



TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

PETTERI NEVAVUORI
MACHINE LEARNING METHOD COMPARISON IN AGRICUL-
TURAL DATA ANALYSIS

Master of Science Thesis

Examiner: University lecturer Pekka
Ruusuvuori
Examiner and topic approved in the
Faculty of Engineer Sciences council
meeting at 27.3.2017

ABSTRACT

PETTERI NEVAVUORI: Machine Learning Method Comparison in Agricultural Data Analysis

Tampere University of Technology

Master of Science Thesis, 74 pages, 13 Appendix pages

May 2017

Master's Degree Programme in Management and Information Technology

Major: Software Engineering and Data Management

Examiner: University lecturer Pekka Ruusuvaori

Keywords: machine learning, deep neural networks, deep learning networks, data analysis, agriculture

The aim of this master's thesis was to compare machine learning methods in clustering and regression tasks with data collected from Finnish dairy farms by Mtech Digital Solutions Oy. Clustering techniques focus on finding similarities between the items of the dataset by examining the data itself. Regression techniques then are used to build predictive models for the dataset. Common theme to all machine learning methods is that they are used to examine data that is manually too complex to handle by applying statistical and mathematical algorithms.

The data has been collected during a timeframe spanning tens of years and has been used by agricultural experts to provide insights and counselling to farmers across Finland on-site. Recent advances in the field of machine learning have however sparked the thesis' employer's interest to employ data-driven modelling and information acquisition practices for standardized and invariant conclusions about the health and progression of farms. There were two datasets formed – one for the clustering task and one for the regression task. The clustering dataset contained information about dairy farms' production and cattle-related health treatment records. The regression dataset then encompassed all the metrics about farms as businesses.

Overall eight machine learning methods were compared, four clustering and four regression methods, respectively. The clustering methods were Hierarchical Clustering, k -Means, Self-Organizing Maps and BIRCH and the regression methods were Ordinary Least Squares, Decision Tree Regression, Multilayer Perceptron and XGBoost. The conclusion for clustering was that k -Means performed the best out of clustering methods, while every method's performance was relatively equal with BIRCH being the only exception. The conclusion for regression method comparison was that XGBoost delivered the best results by performing well score-wise and providing the needed information about most important features.

TIIVISTELMÄ

PETTERI NEVAVUORI: Koneoppimismenetelmien vertailu maatalousdatan analyysissä

Tampereen teknillinen yliopisto

Diplomityö, 74 sivua, 13 liitesivua

Toukokuu 2017

Johtamisen ja tietotekniikan diplomi-insinöörin tutkinto-ohjelma

Pääaine: Ohjelmistotuotanto ja tiedonhallinta

Tarkastaja: Yliopistolehtori Pekka Ruusuvuori

Avainsanat: koneoppiminen, syvät oppivat verkot, syvät neuroverkot, data-analyysi, maatalous

Tämän diplomityön tarkoituksena oli vertailla koneoppimismenetelmiä klusterointi- ja regressioitehtävissä käyttämällä Mtech Digital Solution Oy:n suomalaisista maitotiloista keräämää dataa. Klusterointimenetelmät keskittyvät yhdenmukaisuuksien löytämiseen datasta tutkimalla dataa itseään. Regressiomenetelmiä taas käytetään ennustemallien rakentamiseen. Yhteinen teema kaikille koneoppimismenetelmille on ihmisille liian monimutkaisen datan tutkiminen käyttämällä matemaattisia ja tilastotieteellisiä algoritmeja.

Tutkimuksessa käytetty data on kerätty monen vuosikymmenen ajalta ja maatalouden asiantuntijat ovat käyttäneet sitä maanviljelijöiden neuvontaan ja ohjaamiseen toiminnan kehittämiseksi. Viimeaikainen kehitys koneoppimismenetelmien tutkimuksessa on herättänyt kuitenkin myös diplomityön toimeksiantajan kiinnostuksen hyödyntää dataan pohjautuvia mallinnuksen ja tiedonkeruun menetelmiä standardoitujen ja tasalaatuisten maatiloihin liittyvien päätelmien tuottamiseksi. Tutkimusta varten muodostettiin kaksi datasettiä – yksi klusterointia varten ja toinen regressiota varten. Klusteroinnin datasetti sisälsi karjakohtaisia tietoja sekä tuotannosta että karjan eläinten terveyteen liittyvistä toimenpiteistä. Regression datasetti vuorostaan käsitti lähes kaikki maitotiloihin liittyvät yritystiedot.

Vertailtuja koneoppimismenetelmiä oli kaiken kaikkiaan kahdeksan, joista puolet oli klusteroinnin ja puolet regression menetelmiä. Valitut klusterointimenetelmät olivat hierarkkinen klusterointi, k -Means, itseohjautuvat kartat ja BIRCH. Vertailtaviksi regressiomenetelmiksi valittiin pienimmän neliövirheen lineaarinen menetelmä, regressio-puu, monikerroksinen perceptroni ja XGBoost. Klusterointivertailun lopputulos oli, että k -Means suoritui verratuista menetelmistä parhaiten. Lukuun ottamatta BIRCH-menetelmää kaikki kaksi muutakin menetelmää suoriutuivat verraten lähes yhtä hyvin. Regressiomenetelmien osalta voiton vei XGBoost, sillä se suoriutui hyvin pisteytyksen osalta kyeten samalla tarjoamaan tietoa syötettävien piirteiden painoarvoista.

PREFACE AND ACKNOWLEDGEMENTS

Aihe, jota diplomityöni käsitteli, oli minulle lähtökohtaisesti lähes outo. Vuonna 2015 syksyllä tullessani kouluun ajattelin, etten painottaisi data-analyysiä vaan lukisin ohjelmistotuotannon pitkänä. Toisin kuitenkin kävi, sillä kiinnostus aiheeseen heräsi jo ensimmäisen maisterivuoden keväällä yksittäisen Data Mining-kurssin myötä. Niinpä tilanteen tarjoutuessa tahdoin ottaa diplomityön 30 opintopisteen laajuuden hyvänä mahdollisuutena omistautua uuden asian opetteluun ja hyödyntämiseen. Ja jälkikäteen tarkasteltuna matka on ollut sekä miellyttävä että hyödyllinen.

Vaikka työ on tehty yritykselle ja olen saanut asiantuntijoiden ohjausta, ensimmäisen kiitokseni tästä diplomityöstä ja sen mahdollistaneesta koulutuksesta osoitan Jumalalle. Ilman Hänen työtään Pojassaan Jeesuksessa valmistamansa pelastuksen ja Henkensä jatkuvan rakkaudellisen kasvatuksen kautta en olisi tässä, en liioin edes se persoona, joka tänä päivänä saan olla. Toiseksi tahdon osoittaa kiitokseni vaimolleni Helille. Ilman hänen jokapäiväistä tukeaan ja arkielämästä irrallisten höpöttelyjeni kuuntelua en varmasti olisi saanut kouluani ja tätä työtä tällä tavoin valmiiksi. Seuraavaksi tahdon osoittaa kiitokseni diplomityön ohjaajilleni lehtori Pekka Ruusuvuorelle sekä Mtech Digital Solutions Oy:n Tuomas Loposelle. Yhtä lailla tahdon kiittää myös Mtechin Sinikka Tommilaa monista puhelusta ja annetusta ajasta, sekä TTY Porin henkilökunnasta Jari Turusta ja Timo Mäkistä neuvoista sekä mahdollisuudesta syventää omaa diplomityön tekemiseen liittyvää osaamista joustavasti.

Porissa, 24.5.2017

Petteri Nevavuori

CONTENTS

1.	INTRODUCTION	1
1.1	The Background of the Thesis	1
1.2	Scope and Limitations	2
1.3	Research Methods	2
1.4	Structure of the Study.....	2
2.	MACHINE LEARNING.....	4
2.1	Key Terminology and the Main Principles	4
2.2	Categories for Learning Methods.....	5
2.3	Validating the Trained Model	7
2.4	Risks with Machine Learning	8
2.4.1	Underfitting And Overfitting the Model	8
2.4.2	Curse of Dimensionality	9
2.5	Linear Models	10
2.5.1	The Least-Squares Method	11
2.5.2	The Perceptron	12
2.5.3	Support Vector Machines.....	12
2.6	Distance-Based Models.....	12
2.6.1	Nearest-Neighbour	14
2.6.2	Hierarchical Clustering	15
2.6.3	Distance-Based Clustering.....	15
2.7	Decision Tree Models	16
2.7.1	Classification Trees.....	16
2.7.2	Regression Trees	18
2.7.3	Tree Ensembles	19
2.8	Summary	20
3.	DEEP LEARNING NETWORKS	21
3.1	Core Idea of Deep Learning Networks	21
3.2	A Brief Review of Neural Network History	23
3.2.1	The Back-Propagation Algorithm.....	23
3.3	Neural Network's Key Elements.....	24
3.4	Multilayer Perceptrons	25
3.5	Convolutional Networks	27
3.6	Radial-Basis Function Networks.....	28
3.7	Self-Organizing Maps	30
3.8	Summary	32
4.	MACHINE LEARNING METHOD COMPARISON	33
4.1	Clustering	33
4.1.1	Preparing the Datasets.....	35
4.1.2	Optimal Number of Clusters	39
4.1.3	Hierarchical Clustering	41

4.1.4	k -Means	42
4.1.5	Self-Organizing Maps	43
4.1.6	BIRCH	46
4.1.7	Comparison of Clustering Methods	47
4.2	Regression	55
4.2.1	Preparing the Dataset	56
4.2.2	Ordinary Least Squares	57
4.2.3	Decision Tree Regression	60
4.2.4	Multilayer Perceptron	62
4.2.5	XGBoost.....	63
4.2.6	Comparison of Regression Methods	66
5.	CONCLUSIONS.....	68
5.1	Clustering	69
5.2	Regression	70
5.3	Future research suggestions	71
	BIBLIOGRAPHY	72

APPENDIX A: SQL-Query for Classification Dataset

APPENDIX B: Python Notebook Code for Regression Dataset

APPENDIX C: Kruskal-Wallis H-tests for Production Variable Dataset's Clusters

APPENDIX D: Kruskal-Wallis H-tests for Health Treatment Variable Dataset's Clusters

LIST OF ABBREVIATIONS AND SYMBOLS

ADAM	Adaptive moment estimation
ANN	Artificial neural network
BP	Back-propagation
DLN	Deep learning network
DNN	Deep neural network
HC	Hierarchical Clustering
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
KPI	Key performance indicator
MLP	Multilayer Perceptron
MSE	Mean Squared Error
OLS	Ordinary Least Squares
PCA	Principal component analysis
RBFN	Radial-basis function network
WCSS	Within-cluster sums of squares

1. INTRODUCTION

When viewed from the perspective of computational intelligence, we, the descendants of the human kind, have an extraordinary innate ability to process, identify and classify even the slightest audible and visual cues almost effortlessly and with precision. While computers quickly excelled at tackling straight-forward rule-based calculations, the recognition of spoken words and faces within images proved to be a true challenge to them until in recent thirty to forty years. There have been attempts at trying to describe rules and concepts for recognition explicitly, but a truly successful concept and a focus of this thesis is the concept of artificial intelligence deep learning – especially the use of deep learning networks and the comparison of them to traditional machine learning methods. The comparison of the methods is two-fold with the division to clustering and regression method comparison. Thus, the research questions for this thesis are the following:

1. *“What clustering method provides most intuitive model for farm configurations?”*
2. *“What regression model performs best in predicting and explaining select target variables derived from farms’ metrics?”*

1.1 The Background of the Thesis

Mtech Digital Solutions Oy own an extensive amount of records about the Finnish agriculture from the past 30 years. Until recently the company has strived to satisfy their customers, the individual agricultural entrepreneurs, by looking at the entrepreneur’s farm’s history and interpreting the possible explanatory factors and predicting the future development using individuals’ varying experience and expertise. This also directly translates to the requirement of having to physically locate the agricultural experts directly around the country and have them present their information and knowledge at site.

These are among some of the reasons why Mtech Digital Solutions Oy have started seeking alternative ways for information and knowledge provision to their customers in agricultural sector. One of the ways they have sought is to combine the tools of data mining for both predicting certain key performance indicators (KPI) and finding underlying associations and patterns within existing data. The company doesn’t have much in the form of existing applications, why the seizing of this area of expertise and applications must be started from the preliminary evaluation of applicable machine learning methods.

1.2 Scope and Limitations

The aim of this thesis is to first provide a sufficient conceptual framework for understanding the use scenarios for several types and methods of machine learning and neural networks. The intricacies of the mathematics underlying in the plethora of methods is therefore strictly ruled out and the focus is on opening the core concepts and intuition of the methods. For more interested reader, the bibliography is a good starting point for gaining deeper understanding about machine learning ([1], [2]) and Deep Learning Networks ([3], [4]). The theoretical portion of the thesis introduces and expands upon machine learning methods that are not assessed in the concrete research portion. This is to provide a sufficient bird-eye view about frequently discussed machine learning topics to the reader.

The limited agricultural data with which the research was conducted was supplied by the employer of the thesis. The data was provided in raw form as an SQL-Server database backup and had to be fetched and filtered for the compilation of datasets, and therefore any dataset-related queries or code-based applied processes deemed important for purposes of review or reproducibility are attached as appendices to this thesis. While not every introduced machine learning method was compared for the sake of constraining the broadness, the aim was to pick a method from any main family of machine learning.

1.3 Research Methods

The theoretical portion of this thesis (Chapters 2 and 3) is solely based on reviewing the literature about the subjects of more traditional machine learning and Deep Learning Networks separately. The main portion of the referenced material is derived from books written about machine learning, pattern recognition and neural networks. This material is then reinforced with articles and current proceedings about the subjects when deemed necessary or value adding to the discussion.

The latter practical part of the thesis is then all about using Python as the operating framework for comparing implemented machine learning methods to each other. While the theoretical portion is largely referenced, the practical portion is more akin to reporting the process and findings for clustering and regression.

1.4 Structure of the Study

The introductory first chapter lays the foundation for this thesis by introducing the employer of the thesis, Mtech Digital Solutions Oy, and their problem of being unable to mine information from their accumulated storages of data. The research scope and methods are also expressed.

Chapter 2 lays out the theoretical background for the study regarding traditional machine learning methods and most common algorithms. The chapter's aim is to provide the

reader with basic overall knowledge about the use cases and differences between different machine learning methods as well as the key principles and terminology.

Third chapter is concerned with the background and principles of Deep Learning Networks and their different variations and applications. As in the previous chapter, the aim is to familiarize and provide with overall conceptual framework about the subject.

Fourth chapter is dedicated to the practical research portion of the thesis. The chapter is divided into two parts – the first one dealing with the comparison of clustering and the other with regression machine learning algorithms. The preparation of the dataset, used Python-modules and their configurations are all assessed before analysis of the methods and the methods are compared at the end of the respective part.

2. MACHINE LEARNING

This chapter is about traditional machine learning models, their fundamental principles and use scenarios. The chapter begins with an overview into the key terminology of the concept of machine learning. It is followed by division of varying learning methodologies into categories, some of which are more frequent in traditional machine learning methods and the others in neural networks discussed in Chapter 3. After this the subject of validating the model is discussed briefly accompanied with discussion about the risks involved with machine learning.

Then the chapter shifts into introducing several of the most common machine learning models. First the linear models, like least-squares, the perceptron and support vector machines are looked at. Next are the distance-based or clustering methods, such as nearest-neighbour and hierarchical clustering. Before the summary of this chapter the subject of tree models is also touched upon.

References to [1] are frequent in this chapter due to exceptionally well versed and understandable articulation of core fundamentals and principles regarding traditional machine learning methods. The reading of that book is highly recommended for readers looking for more within the subject of this chapter.

2.1 Key Terminology and the Main Principles

The core concept in machine learning is generalization. A well-generalising machine learning algorithm transfers the knowledge it has acquired during training to real, operative situations. A well generalising stock trading algorithm is able predict the stock movements well; a well generalising KPI tracker is able to pass on its accumulated experience to perform forecasts for data it has never experienced before. The knowledge is contained within a model, which could be seen as a set of instructions on how to handle or manipulate data patterns given as input to the machine learning algorithm. A descriptive model categorizes the input patterns to groups with similar features and aims at explaining the differences. A predictive model is used to predict a numerical or categorical outcome of an unlabelled input data pattern. [1], [5]

When the model's prediction outputs are classes, be it binary e-mail spam detection, multi-class vegetation detection from a satellite image or animal kind or even species recognition from images, the model's task is to act as a classifier. If the outputs are real numbers, like production growth percentage, change in sales or number of sunspots, the task of the model is regression. These two tasks require a training dataset, which has a true or desired output for each set of inputs and are included in broader learning method called supervised learning. If training datasets are not available or they're extensively

costly to produce in terms of labour, time or money, the models can also be trained without the correct answers. The machine learning task of taking a certain set of items and using their features to set the items in broader similar categories or finding hidden associations is called clustering. Clustering belongs to broader learning method called unsupervised learning. [1], [3]

The key difference between supervised and unsupervised learning is thus that supervised learning requires a training set with target values while unsupervised does not. From this distinction also stem the broader model categories of predictive model for learning with target values and descriptive model for learning without. Third learning method differing from the already presented two is the reinforcement learning method, where the model interacts with its surroundings and has a feed-back loop between the learning and the experiences. Classification, clustering and regression are only a small subset of a bigger variety of possible machine learning tasks. Other tasks for machine learning include transcription (optical character recognition), machine translation, structured output (parsing sentences to grammatical structures), anomaly detection, synthesis and sampling (texture generation) and denoising. [1], [3]

2.2 Categories for Learning Methods

The main learning categories, as stated in preceding subchapter, are supervised and unsupervised learning. Sometimes additional divisions are made ([4], [6], [7]), sometimes learning methods are handled on a per-model basis ([1], [8]). For clarity, the main division used in this thesis is done between supervised and unsupervised learning methods. Methods directly or closely related to these two are assessed in their corresponding sections.

Supervised learning makes use of teaching the network with interrelated input and target data, or independent and dependent variables. The model utilizing supervised learning method is fed with input data of which the desired outcome is known a priori and the outcome is then compared to the target value. The error is then used to calculate the incremental adjustments to each weight of the network and the process is repeated until desired error margin is reached. This learning method is derived from optimization techniques and can be used for making predictions after the training of the model is done to desired degree. Teacher and student provide good real world parallel to this method. The target is to achieve a definite minimum error or at least point close enough to it within the data. Thus, the biggest stumbling block for supervised learning is the ability to escape from local error minima. [1], [5], [6]

While there exists a multitude of learning models derived from different scientific fields, some notable methods utilizing the concept of supervised learning are the following:

1. Reinforced learning shares similarities with supervised learning method as it also makes use of target data for model output validation. The main difference is that

instead of providing the model with error values, the error output is only a binary pass/fail. This means that incremental weight adjustments are not an option and the strategy for weight adjustment calculations is different. Instead of guided adjustments there's an element of random guesswork to the model's training. It has the same critical stumbling block of settling in a local minimum instead of a global one as the supervised learning model.

2. Delta learning rule makes use of continuous adjustments to the model by measuring the changes or deltas in errors. There is parallelism to supervised learning, but this modifies the model continuously when the other is used to teach the model only to certain degree. The delta rule is also known by the names Widrow-Hoff learning rule and Least Mean Square for it aims at minimizing the mean square error between the model's output and target value.
3. Gradient descent learning rule uses the first derivative or the gradient of the error between the output and the target and then calculates the required adjustments for each weight. The goal is to decrease or descend the error function while avoiding local minima to reach the actual global minimum. This is especially desirable, because local minima are usually frequent in wave-like data. Deep learning algorithms are usually based on this algorithm. [1], [3], [5], [6]

Unsupervised learning is an opposite to supervised learning in that there are no target values to conform to and is sometimes referenced as self-organized learning and clustering. The machine learning model is fed with multiple different input patterns which are then categorized randomly. Through training the model learns to classify sets of inputs to distinct categories. While there's no target data, the model needs some guidelines for categorization to be successful. This learning method is good for classification. A good parallel is a person that's asked to divide a set of objects to two categories. How the person does it is his or her personal choice, but the division is usually achieved nonetheless. [1], [3], [6]

Notable learning methods belonging to the family of unsupervised learning include at least the following:

1. Competitive learning uses multiple outputs that compete to provide the model's output for any distinct input data pattern. There can be multiple dominant output neurons for the whole dataset, but only one produces the output for each set of inputs in a winner-takes-all-fashion. In other words, the model's outputs are trained to respond to different input patterns. A team of experts with distinct fields of expertise could be used as a parallel for this learning method. While competitive learning can be used with supervised learning [6], it can also be used in an unsupervised setting. [4], [9]
2. Hebbian learning is also a learning method frequent with self-organizing neural networks. The main principle is to mimic the actions of two closely and repeatedly

communicating neurons that have their communicative efficiency further increased as time passes. In this model, the powerful connections are even further empowered. In contrast, the Anti-Hebbian learning actively avoids this kind of strengthening of connections. [6], [10], [11]

2.3 Validating the Trained Model

Regardless of the category of the machine learning task, the algorithms in machine learning have one common goal for them – to search for an underlying structure, a pattern inherent in the dataset at hand. The key to successful utilization of machine learning is find the most suitable algorithm in conjunction with the right features in the data while utilizing the most suitable learning methodology. The accuracy of the model can be best validated by dividing the training set to training and test set. While the model might seem to cope well with the training set reducing the error levels to a desired minimum level, the test set might prove the model to perform poorly with data it hasn't been exposed to yet. Cross-validation is training validation process, that further ensures the fitness of the model. The training data is separated to for example ten parts of which nine parts are used for training and the remaining part for testing. The training is then repeated as many times as there are segments of training data in a way that every part is used once as a test set. The measurement indicating the performance of the model is the mean value of all test set performances. This process works well only for supervised learning though. [1], [3], [4]

Another way of looking at the validation problem is the statistical point of view regarding to varying outcomes of different situations. In cases, where the input dataset is sparse and the population from which the model is thus built upon is small, there might be significant variations between the outcomes of different trained models. This in turn would render any single trained model questionable to say the least. A way to overcome this is called bootstrap aggregation. To illustrate this, suppose that there was a single non-repeatable test conducted. This could be a medicinal test, or a feedforward neural network with randomly initialized starting weights. The outcome of the test was 10 positives out of 100 negatives and the question is: Would consecutive tests done with the same parameters yield comparable results? Bootstrap aggregation, or just bootstrap for simpler referencing, would in this in case use the sample pool of 10 positives and 90 negatives and pull out a hundred random samples with replacement (meaning the sample pool would always have the same number and ratio of samples). This would be then repeated N times. This in turn would give information about the standard deviation about the outcomes and provide tools for weighing the test's outcome [12]. Bootstrap can be used with classification tasks

belonging to the family unsupervised learning by comparing the outputs with knowledge of approximated distribution about the input data [13].

Unsupervised algorithms are however tricky to evaluate from the performance viewpoint. The nature of unsupervised learning is that there is no ground truth on which the training is made on and to which the results can then be compared to. There are however techniques that measure the goodness of clustering by looking at cluster densities and the separation of clusters. One of these is the calculation of Silhouette Coefficient. The core concept is that every cluster is represented by a silhouette with distinct separation and tightness (or density). The Silhouette Coefficient is calculated by using the mean distances of samples to their clusters and their nearest cluster. The closer the coefficient is to 1, the better and the closer to -1 , the worse the clustering is, thus the output is confined to a certain range [14], [15]. Another similar clustering performance evaluation tool is called Calinski-Harabaz Index, which computes a score for the model by defining the ratio of dispersion between points within a cluster and points between clusters. Higher score translates to better defined clusters, but the score's range is not confined [14], [16].

2.4 Risks with Machine Learning

While machine learning models offer insights into data that might be concealed to a human agent, the models are still formed only on principles governing the machine learning method at hand. Because machine learning methods are after all only programmatical implementations of mathematical and statistical concepts, there are some risks that are good to know and recognize.

2.4.1 Underfitting And Overfitting the Model

Two particularly important terms related to the goodness of the model are overfitting and underfitting. In both cases the terms refer to the performance of the model related to the data. A good example of model's fitting is forecasting the trend from a set of data points. An underfitting model is unable to reveal the underlying pattern of the trend and it might just give out a straight line indicating the direction of the trend but completely missing the characteristics of it. This could be due to choosing a machine learning model that is too simple for the dataset, meaning the number of parameters of the model is relatively speaking too minuscule. An overfitting model in the other hand accounts too well for the noise and variation that it fails to provide an explanatory trend underlying the dataset. Overfitting can occur when the machine learning algorithm is too complex for the set of data, meaning the contrary to the number of parameters in underfitting. [1], [5]

Imagine you had a dataset with its data points originating from an exponential distribution with some added noise. Underfitting model is unable to see the gently sloping beginning and the steepening climb as observations progress. Overfitting in the other hand goes in

to the extremes at both ends of the dataset. Both ill-fitting models fail recognise the underlying exponential structure and progression in the data. A visual example of underfitting and overfitting is depicted in Figure 1, where the dots denote the ground truth and the background contours the decision boundaries for the trained model. In both border cases the ability to predict correctly or classify in an elegant and meaningful way is diminished due to under- and overfit, while the middlemost model fits well to the dataset.

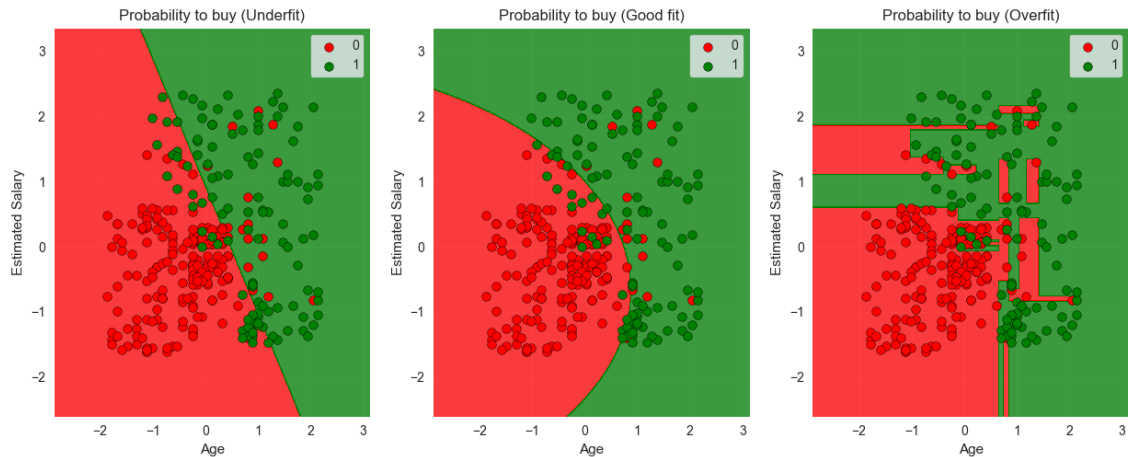


Figure 1. Examples of under- and overfitting models failing to segment the probable buyers in elegant way.

Some models are inherently more prone towards underfitting (linear models) and some towards overfitting (tree models).

2.4.2 Curse of Dimensionality

When dealing with data instances that have multiple variables and therefore are high-dimensional, the data tends to be sparse and even extremely so. Every point of data then has a tendency of being far away from any of the closest neighbouring data point thus rendering the information about the distance uninformative. While excessive number of features is one of the more certain ways to get afflicted by the curse, the incorporation of irrelevant features also helps in contracting it as well. The aim of using machine learning algorithms with datasets is, as discussed earlier, to search for underlying patterns and associations within the data. Exposing the algorithms to features irrelevant to the research subject can add so much noise to the learning process that the meaningful relations are lost in it. There exist some tools for reducing the dimensionality of the data, two of which are feature selection and principal component analysis. The latter of the two is a well-known feature construction method. Principal components are essentially newly formed linear combinations of given variables. An example of this would be the utilization of matrix decomposition to find genres of movies within a movie database. [1], [2]

2.5 Linear Models

Linear models are part of bigger family called geometric models. Geometric models work often under the assumption that the input instances are formed of variables having real number values. In the context of machine learning these variables are often referred to as being the features of an instance. This means that the space in which each input instance is modelled in has as many dimensions as the instance has features. To clarify, an instance could be a bovine farm. All the information relevant to that farm would constitute the set of features. The number of bovine would be one dimension. The production of milk another. And as many as there are features, as many there are dimensions to that single instance. And this set of dimensions forms the instance space in which the linear models operate.

Linear models make use of interpreting geometric elements like lines, planes and hyperplanes as decision boundaries for data separation and data classification. Linear models are based on well-understood mathematics and provide a simple and interpretable way of analysing the data. This is almost orthogonally contrary to Deep Learning Networks, where the interpretation of the models by their parameters requires extensive expert knowledge and is still an on-going research subject [17]. Linear models are parametric, in that they incorporate elements that must be learned from the dataset (e.g. weights), and the number of parameters is relatively small. This makes them less prone to overfitting and more stable towards small variations in the data but can sometimes lead to underfitting of the model. The family of linear models have solutions for all kinds of prediction-related classification, probability estimation and regression tasks and a simplified example of linear regression is given in Figure 2. [1], [18]



Figure 2. Two-dimensional linear regression of an imagined dataset.

2.5.1 The Least-Squares Method

Least-squares method is a linear model suitable for regression and classification. The method's core is finding an estimation for the learning function so that the sum of the squared errors is minimised. If data is represented in a 2D Cartesian coordinate system, in concise terms the method finds the best function for a line that has the smallest squared distance to each of the points. This also means that the method is quite sensitive to outlier points, especially in smaller datasets. There are variations to this method as well. Ordinary least-squares assumes noise only in y-direction, while total least-squares assumes noise in x- and y-coordinates. The combination of linearity within a coordinate system makes also a good basis for binary a binary classifier. [1], [18]

While least-squares tackles with single values and two dimensions at first glance, the method works well also for matrices and vectors. This is called multivariate linear regression. Implementing multiple variables or features to least-squares method requires an assumption of independence between the features. With that assumption and matrix operations it possible to break the multivariate linear regression problem in to as many univariate linear regression problems as there are columns in the original matrix. [1], [18]

2.5.2 The Perceptron

While perceptrons are a subset and in a way the originating point of neural networks, they also work well as linear or binary separators for classification. The form of perceptron's output is reliant on the chosen activation function, which can be of hard-limiting or e.g. logistic regression type. Perceptron's functioning principle is to iterate over a training set of data containing desired target values and modifying the weighing of inputs until the difference or error between output and target values is minimized. The outcome and the method of classification is very similar to the least-squares method. The perceptron is trained with every single point of data in the dataset. During the training process, the weighing factor is adjusted to minimize the output error. In this way, the perceptron finally describes a line separating and thus classifying the data. [1], [2], [18]

The downside of using a perceptron for classification is that data must be linearly separable. Otherwise the perceptron will not converge and will try to do so ad infinitum. On the positive flipside, the perceptron can be harnessed to provide e.g. online classification in a way that updates the classification model when a misclassification occurs. [1], [2]

2.5.3 Support Vector Machines

One of the peculiarities of linearly separable data is that there exists an infinite number of ways to make the separation. Some of the separations provide greater margins than the others, but the point still stands. The margin in linear separation refers to the distance of closest data points of differing classes to the linear separator, the line (or the line function, to be more precise), which is also known as the decision boundary. Support vector machines are a tool for determining the linear separator with largest equal margins for closest differently classified data points. The training points closest to the decision boundary are called support vectors, from which the name of this linear model also stems. Support vector machines are essentially a classifying optimization tool. [1], [2]

On the contrary to perceptrons, support vector machines are less sensitive towards linearly inseparable data. This is enabled via the use of slack variables to each training data point and the allowance of margin errors – which is called soft margin support vector machine. [1], [18]

2.6 Distance-Based Models

Machine learning models that operate on the principle of forming groups of data points belong to the family of distance-based models. The core idea of all of them is clustering. Clustering of data in terms of data mining and analytics translates essentially to searching for subgroups within a larger main group of data points, the first of clustering's key elements. The other key elements of clustering are the space to which the points belong to and the distance between the points. Clustering focuses on finding similarities between

the items of the dataset. The means of clustering can be divided to two fundamental strategies:

1. **Hierarchical clustering** considers every point to be a single cluster at the beginning and then expands the clusters to contain the neighbouring points. The way the neighbours are defined varies, but the main idea still holds. The clustering is considered finished when a termination clause is satisfied – total number of desired clusters is attained or further clustering would be done on the expense of cluster compactness, to name a few examples. This strategy is more viable in relatively smaller datasets. In short, the clusters are expanded from single points.
2. **Distance-based clustering** assigns the points to clusters by examining them in some predetermined order and a ruleset for cluster matching. This method could also be called point assignment clustering. The number of clusters is first given an estimation contrary to hierarchical clustering and points can get swapped between clusters or even left out as outliers in the process. K-means algorithm is one of the best-known point assignment algorithms. In short, the points are assigned to somewhat pre-set clusters. [1], [2]

Without going in to too much detail, there are several ways providing differing results for point-to-point distance calculations. There's Minkowski distance, Euclidean distance and Manhattan distance to name a few. These different ways of calculating the distance propose each its own way of handling the distance between the 2D-Cartesian coordinate system's main axes (x- and y-axes). For example, the Manhattan distance of 1 is 1 along the axes, but forms direct lines to these points for off-axis points. This means that for points $(x_1, y_1) = (0.5, 0.5)$ and $(x_2, y_2) = (0, 1)$ the Manhattan distance from origin is 1 for both. Other methods of distance calculation form handle the distance in more circle-like and even increasingly square-like fashion. In Figure 3 the distance to point (1,1) would be two Manhattan distances (red diagonal rectangle), while being only one Chebysev distance (blue square). The selection of distance calculation method has an effect on the outcome of clustering, as different methods assign different distances to points not along the same vertical and horizontal axis. [1], [18]

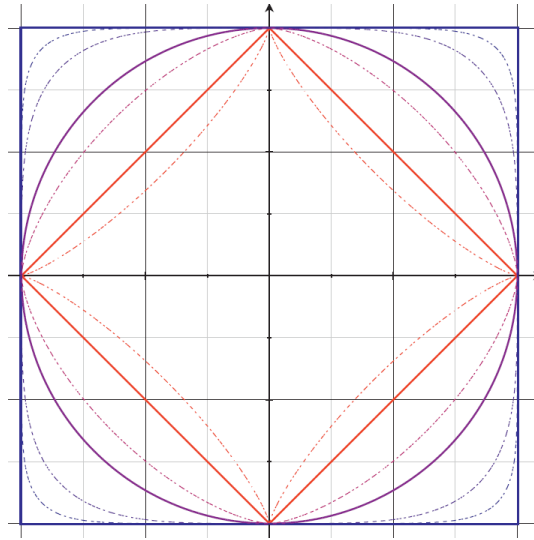


Figure 3. Visual representations of different distance measures used in distance-based models. [1]

Clusters provide often useful data about grouping of points and thus the underlying structure of the data. If movies were the points, the clusters could be the genres. Clustering also eases the calculations by allowing the researcher to use cluster central points called centroids (or even clustroids for extensive number of original dimensions and clusters) for cluster-related investigations. One way to look at centroids is to view them as centres of mass for distinct group of data points. [1], [2]

2.6.1 Nearest-Neighbour

Nearest-neighbour classifier is one of the simplest distance-based classifiers. It is the most commonly used and its simplicity is similar to that of univariate linear regression. Nearest-neighbour can be used to solve a multitude of machine learning tasks, as it is quite oblivious to the type of the target variable during the training. The final output can be real numbers, text or even video. Nearest-neighbour classification method memorizes all the instances of the training set, which then enables the method to separate the classes perfectly from the training set. As nearest-neighbour is a subset of point assignment clustering, the process of classification requires choosing of training points representing different classes. Because the method forms the class boundaries strictly out of the training data, the method has a risk of overfitting the model if the dataset is limited or noisy. Because nearest-neighbour finds the neighbours of each point, the increase in dimensions affects the method's performance in the way of curse of dimensionality. [1], [2]

One of the most famous nearest-neighbour methods is the k -nearest neighbour. It is essentially a voting system, where the classifier votes between k nearest neighbours of the point and predicts then the majority class. The value for k can be anything greater than one but still not greater than the number of training points. This is because when k starts closing in on the number of training points, the predictions start getting more uniform and

the method starts losing the ability to predict different classes. A workaround exists to this though and it involves incorporating distance weighting to the voting system – the farther the voter, the lesser the impact of the vote. [1], [2], [18]

2.6.2 Hierarchical Clustering

Hierarchical clustering uses the distance of data points to finally construct hierarchical trees or dendrograms. Because this method uses the distinct features only indirectly and rather considers every point in relation to the whole instance space, this clustering method is descriptive. Descriptive clustering means that it aims at representing associations and as a learning method it is considered as an unsupervised one. To open the notion of the dendrogram a bit more, a tree-like dendrogram has every data point of the dataset at its leaves and binary branches in between them. Binary branching means that when coming from the level of every data point encompassing broadest description to the level of distinct data points, the descriptive accuracy increases and the amount of grouping clustering decreases. The optimal clustering tends therefore to lie at a point between the distinct points (also termed a level 0) and the highest all-encompassing description level. Usually though the user is the one making the decisions about the desired number of clusters and therefore also determining the descriptive broadness of the clusters in relation to the data points. Hierarchical clustering and the dendrogram were both used in the clustering method comparison found in Chapter 4. [1], [18]

2.6.3 Distance-Based Clustering

Distance-based clustering revolves around the concept exemplars. Exemplar, as a word, is defined as representing the one that serves as a model or an example [19]. This definition gives leeway to the earlier defined methodology of using a set of pre-determined data points as exemplars of clusters. To put in other words, some points are cherry picked first from the data that are thought to represent distinct subgroups within the larger dataset. After this the distance to these exemplars is used as a precondition for assignment to distinct cluster. This also means, that this clustering method doesn't operate by searching for associations but rather providing a set of instructions using different measuring methods for predicting the correct cluster for the data point. [1], [18]

K-means problem, that is also directly associated to the K-means algorithm, relates to searching for the partitioning of clusters that minimizes the scattering of data points within the clusters. It is a problem related to compacting the clusters, where K stands for the varying number of cluster means and vectors. There exists no efficient solution to this problem (being an NP-complete problem), thus requiring the use of a heuristic algorithm to find a solution to single instance of the problem. K-means algorithm is one attempting just this as it tries to find a point where the scattering is minimized within the clusters. There can't however be absolute certainty about the minimum – it might not be the global

one. There are also variants to this method, like K-medoids. K-means clustering was used in the clustering method comparison found in Chapter 4. [1], [2]

2.7 Decision Tree Models

Expressive and easily understandable, the tree models are amongst the popular machine learning models. Tree models have a wide variety of use cases from classification to image recognition, regression and clustering and are sometimes abbreviated as CART (classification and regression trees). Tree models commonly use binary branches for distinguishing each testable feature and allow chaining of features with some divisive boundaries. Tree models can be also expressed in terms of propositional logic, which makes the developed models easily decipherable for humans. Tree models commonly make use of a feature tree structure. Each internal node of the tree is a feature, edges are labelled and leaves represent logical expression. A set of branching labelled edges emanating from a node is called a split. The chained expression and labels then constitute a hypothesis space, in which the data is the processed. The classification process is centred around rejection of non-acceptable class-definitions until finally an accepted one is reached. [1], [18]

To use trees as machine learning tools a set functions need to be introduced. First one is the test for homogeneity. This is to test if the instances in the dataset are similar enough. The next one is the labelling function. Homogenous instances can be labelled similarly. If, however, the distinct data points are not homogenous, there must be a way to differentiate between the instances of dissimilar nature. This is done using a splitting function, that splits the instance to give out the proper ranges for the two heterogenous groups. This is where also the growing of the tree happens. In short, the data is divided into subsets of data per similarities between the data points. As this method is of greedy nature, the selections for splits and groupings is done only on available information thus carrying with itself the possibility of resulting in comparably inferior model in terms of generalization – risking overfitting, in other words. [1], [8], [18]

2.7.1 Classification Trees

Decision tree models are especially concerned with making best splits among the features in a dataset, in other words grouping the instance space to proper segments. The models use the measure of impurity for determining if a feature is a good splitting factor for a given data set or not – the greater the impurity of the observable feature, the lesser the fitness for it to be used as a splitting feature. Purity and impurity are all about the distribution of instances within the data or instance space. A feature is considered pure, when the instances belong distinctly to one of the two child nodes branching from the parent feature node. A split between Boolean true or false values would be a pure split. An impure feature would then be a feature, where the child nodes would have relatively large

overlap between them, making the partitioning of instances with the impure feature not self-evident. The impurities can be compared both locally between the parent node and branching children and globally to each other nodes' impurity. Impurity calculations can be done by using the minority class, Gini index or entropy functions and they are always related to the decision of whether to split or not to split an existing leaf to a new leaf. [1], [18]

While decision trees are good for classification, ranking tree models can be used to further rank the segments in some order and are in effect an extension to the decision trees. Because decision trees already have the information about the features' local distribution, the ranking of features can be achieved by arranging the leaves looking at the magnitude of positive space they cover. The more, the higher ranking and vice versa. The ability to rank the features is also property of decision tree models with some grouping models included, but no traditional machine learning models providing grading can achieve this. [1]

To provide a classification example, let's use an imagined dataset about customers classified by buying potential with age and estimated salary as only input features. The standardized ground truth values are given in Figure 4, where a green dot denotes the customer buying the product and a red not buying it. Originally the dataset is from an online machine learning course example, but modified to fit the purposes of presentation.

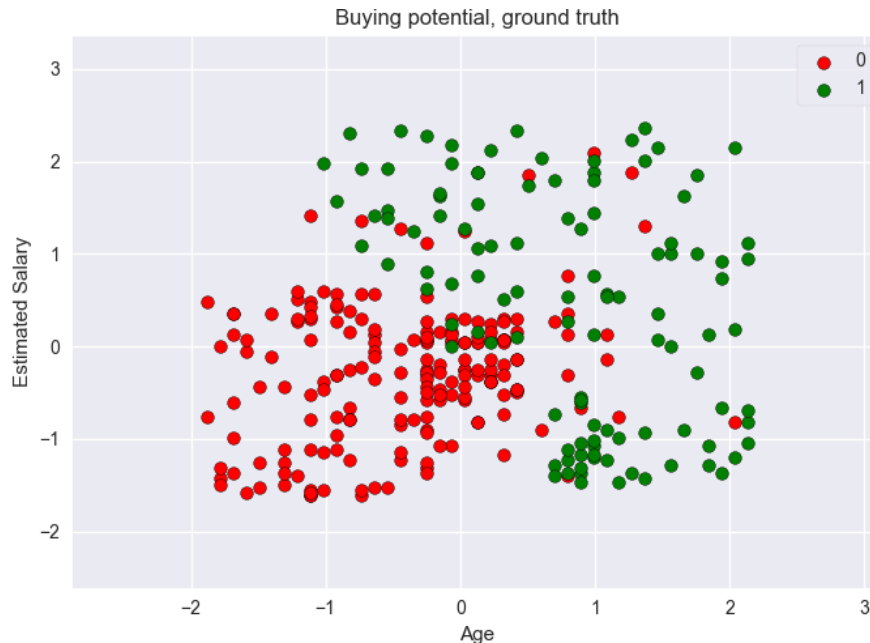


Figure 4. Ground truth for decision tree classification example dataset.

With only two features, the structure of the trained tree given in Figure 5 grows quickly in levels and complexity. While the values and texts are too small to read, each binary split is done according to a defined feature-related threshold level. The first split is related

to age – if age is below 44.5, go left (true) and otherwise go right (false). Second level is then about salary and the third is already a totally mixed level.

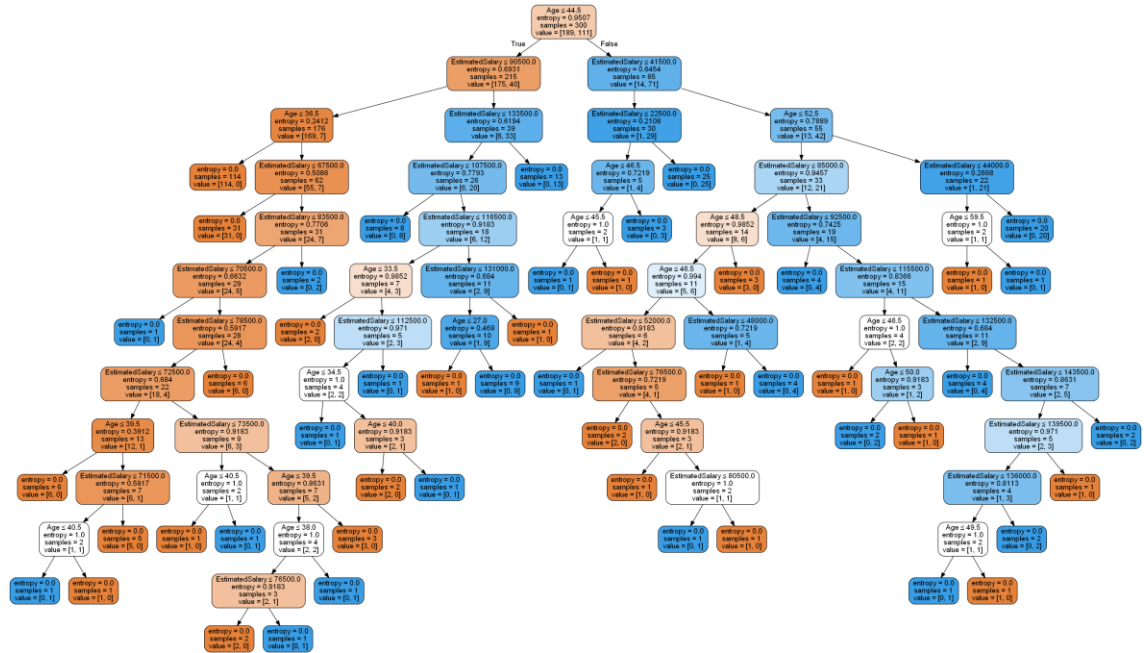


Figure 5. Structure of a trained classification decision tree with example customer potential dataset.

The structure of the tree denotes multiple leaves, which in turn translates to multiple classification regions and possible over-fitting. In fact, the rightmost figure shown in Figure 1 under already discussed under- and overfitting of the models is the visualization of this decision tree.

2.7.2 Regression Trees

Tree models can also be used for regression tasks. This can be achieved by taking a different point of view towards the impurity and using it as variance calculator for a feature. This then helps in determining the basic properties of featural distributions, which helps in handling the features as continuous variables rather than binary ones – which is the most crucial difference to classification trees. Other than this, the branching is however very similar to the classification trees. For regression tasks the aim is to minimize the overall variance of the tree's features, which then can be used as a regression model for the input data. Another way to achieve tree-like regression model is to use linear models in the leaves of the tree. Regression trees are prone to overfit when the data is too sparse or the dataset is small. As opposed to the linear models with continuous values, the regression decision trees produce a step-like model where each leaf is effectively a range of continuous values that produces a single output. An overly simplified example of this is given in Figure 6. [1], [8]



Figure 6. An example of decision tree regression with an imagined dataset.

2.7.3 Tree Ensembles

As already stated, tree models tend to learn the training dataset too well, e.g. they tend to overfit. An intuitive way to tackle this is to employ model combining. By combining multiple similar or varying models and for example averaging the outputs for a single output, the result is usually more accurate than using just a single model. These kinds of combinations of regression and classification models are known generally by the name model ensembles, and are used especially with tree models, i.e. random forest method. Random forest approximates the output by using multiple tree models and then produces the average output of all the trees. While the number of trees in random forest method can be in tens, the preferred number of trees is in hundreds. The focal point of the approach is to induce variance in the models, which is achieved with bootstrap aggregation or bagging. Bagging is effectively about constructing a random sample using the original dataset as the base for distribution's parameters. [1], [12], [20]

Another way to employ ensembles is to use them to boost the ensembles models. Instead of using multiple models in equal manner for an averaged output, boosting uses subsequent models to enhance or boost previous models. This way the process of learning turns into an iterative one. An illustrative example would be that the use of bagging with ensembles is like taking a bunch of runner to a track and calculating the average time. The use of boosting then would like a relay, where the next model in the ensemble continues from where the previous finished. A recent implementation of boosted ensembles of trees is known as XGBoost, which makes use of gradient boosting and trees. An extended introduction into gradient boosting is provided in [21], but the core principle is the one

described already – to fit consecutive models to improve the overall output subsequently. [1], [22]

2.8 Summary

This chapter introduced the reader to key terminology in machine learning and the high-level division of machine learning methods to supervised and unsupervised methods. The chapter assessed also methods for model validation for both high-level machine learning methods. Risks, like the tendency to ill-fit and the curse of dimensionality, were also assessed.

Supervised methods require information about the ground truth to which the trained models are weighted against and the validity of the models can also be assessed with comparisons to ground truth. Traditional supervised methods include linear and decision tree models, like the least-squares linear method or decision tree classification method. Unsupervised methods in the other hand have no information about target classes or values and focus on dividing the dataset by similarity measures. Traditional unsupervised methods include distance-based methods such as nearest-neighbour and hierarchical clustering.

3. DEEP LEARNING NETWORKS

Deep Learning Networks (DLNs), Deep Neural Networks, Artificial Neural Networks, Neural Networks – these some of multiple names given to one and same model paradigm in its phases of development. While the mimicking of biological neural networks was the original starting point of the concept of neural networks as computational units, they still belong to the broad spectrum of machine learning methods. The reason for taking the subject apart from more traditional machine learning concepts, such as those introduced in the previous chapter, is the fundamental difference in the design and the paradigm of the key concept – learning, adaptation and plasticity and parallel processing of input data.

Therefore, in this chapter there will be first a brief introduction to the idea behind deep learning networks, which will then be followed by also a brief overview of the history of the paradigm. Then the key elements of neural networking are introduced after which a selection of important neural network models is represented. Again, it is noteworthy that the aim is to familiarize the reader to the concepts and not delve deep in to the underlying mathematics. Also, it is to be noted, that the terms DLN and neural networks are used rather interchangeably and refer to same thing. The term artificial neural networks is quite common in the literature, but it has been deemed better to use terms that are up to date and used in modern research.

This chapter advances by first introducing the core idea behind the neural network paradigm, which is then followed by a historical review. The key elements are then introduced, after which the chapter follows in the footsteps of the previous chapter by delving a bit more deeply into some of the most common neural network models. These are the multilayer perceptron, the convolutional network, radial-basis function networks and self-organizing maps. Lastly the chapter is summarized.

3.1 Core Idea of Deep Learning Networks

The process of discovering the basic principles governing the workings of the brain has been one of several millennia, but only just the last century was the one witnessing the advent of true research in to biological and artificial neural networks. The idea of DLNs is firmly rooted in the study of biological neural networks of which the human brain is the prime example. The human brain is designed and created in such a way, that it can process information at an astonishing speed and accuracy even when the relevant information is buried under heaps of noise. It has been estimated that the average has the maximum operational capacity at $10 * 10^{15}$ operations per second, setting it at the peta-scale [6]. In computer-equivalent terms, the petaflops-scale computational capacity has been achieved only during the last ten years or so. The acronym ‘flops’ comes from the words

floating-point operations per second. The last listing of top 500 supercomputers and their records from November 2016 shows that the top ten systems have and the following operating statistics:

- Average number of cores: $1,84 * 10^6$
- Average maximum petaflops: 22,6
- Average power consumption: 5960 kW. [23]

To gain perspective, a single brain has an average volume of one third of a litre, weighs approximately 1,5 kilos and incorporates around $100 * 10^9$ neurons with an average of a thousand synaptic connections to other neurons. The power consumption is shared with muscles and intestines in a system that requires an average of 2200 kcal or 2.5 kW of energy input daily. The architecture and design is therefore a marvel on many levels. And the brain isn't the only network-like architecture in biology. The eye is also a complex system of retinal neural networking. [6]

The original neural network research's aim was to mimic the neurons and their connections to gain brain-like efficiency in computing. The modelling of these networks was achieved using mathematical descriptions that attempt to describe the behaviour of neurons and their networks. It is worth mentioning though, that the mathematics used do not and even cannot describe the functions of biological brain exhaustively for mathematics is after all only a human invention for describing the natural phenomena. The basic attributes of neural networks are the architecture of the network and neurodynamics. The architecture describes the structure of the network, the number of neurons and their connections to each other. Processing in successively complexing stages has an origin in the retinal neural network and the way the visuals are processed. From this stems the layered overall neural network architecture. The neurodynamics then defines the functional properties of distinct neurons in the system – how they learn, associate and compare incoming information to existing knowledge. [4], [6], [24]

The main difference to sequential algorithms is that neural networks use parallel decomposition to break complex information structures in to smaller and easily manipulatable fundamental elements. The idea to this comes again from the brain. The brain, when processing an image, doesn't record every single distinct colour variation of the picture but rather the characteristic and elemental features of shapes, colours and their relationships. In the same manner, the neural networks work to first identify the distinct features and then forming a more concise overall picture of the observation. It is also noteworthy, that the artificial neurons are *artificial* – they do not even closely attempt to describe the biological neurons. Artificial neurons draws from their biological counterparts mainly in the ways of weighed interconnectivity but the inner workings of the biological neurons is a subject of different thesis. [6], [24]

3.2 A Brief Review of Neural Network History

The idea of a single artificial neuron derived from its biological role model was presented by McCulloch and Pitts as a mathematical model in 1943. This is now known as McCulloch-Pitts model and it was a simple model with no feedback for adaptation or learning. It was the first stepping stone in the line of development of first DLNs. The idea of McCulloch and Pitts was then expanded to pattern recognition. The idea was that all input values are multiplied by weights, the resulting values were summed and then passed through a non-linear activation function providing for example a binary output of either 0 or 1. This very similar to how single neurons operate in modern neural networks as well. [6], [24]

The need for feedback was however quickly acknowledged and this acknowledgement gave way to development of the perceptron by F. Rosenblatt. The perceptron made use of target values for output comparison in the training process. The learning mechanism depended on the difference between the output of the neuron and target value – the error. The drawback of this model was that every distinct set of input patterns required a separately taught perceptron. The original perceptron also failed to pass the Exclusive OR (XOR) function test, where similar inputs (00 and 11) give an output of 0 and dissimilar (01 and 10) an output of 1. Although criticism ensued, the latter deficiency of failing to produce the XOR function's outputs was handled by developing perceptrons utilizing back-coupled error correction mechanism. [6], [24]

After the development of a single perceptron, the next transition was to use many perceptrons at the same time in the same calculating system. The multilayer perceptron formed a topology similar to a simple neural network topology – there were layers for input neurons, hidden neurons and output neurons and the number of layers and neurons between input and output layers wasn't fixed. There exists a variety of viable learning algorithms for MLPs but two from the common end of the spectrum are the Delta and Back-Propagation learning algorithms. [6], [18], [24]

3.2.1 The Back-Propagation Algorithm

Back-propagation algorithm developed during the 1970's has been widely used in multilayer neural networks. With MLPs there is generally an increasing difficulty in calculating the adjustments to each individual weight with the increase in hidden layers. While at the output layer, the error is a straightforward difference between target and output values, the impact of each weighted single neuron's output in hidden and input layers to final output error requires a more complex mathematical solution. While the feedforward consists of relatively easy tasks of multiplying the inputs with their corresponding weights, summing all the weighted inputs for each successive neuron and then passing the sum through an activation function all the way to the last output neuron, the weight adjustment process is quite harder and more resource intensive. The back-propagation makes use of

calculating the partial derivatives or gradients for each weight impacting the successive neurons regarding calculated error and then updates each weight's multiplier accordingly. It is also an application of statistical technique known as stochastic approximation [4]. While being able to provide very accurate results [25], the algorithm has also received notable criticism for its calculation-intensive operating principles resulting in slowness of teaching. The number of gradient calculations is proportional to number of weights in the neural network and the adjustment calculations are done with each input-output-pair. This results in poor performance in real-time applications. [6], [18], [24]

3.3 Neural Network's Key Elements

When speaking in terms of neural networks, there are some key elements to the artificial neurons. From here on words *neuron* and *artificial neuron* are used interchangeably. Inputs represent the data from the origin or preceding layers of neurons. Before reaching the neuron, every input is multiplied by an input-distinct weight, which represents the strength of the connection between the neuron and the connected preceding data source. Upon reaching the neuron, every weighted input is then summed and pass through a non-linear activation function. The activation function can be thought of as a scaler or classifier of sorts. Logistic activation function for example scales the output of a neuron to the range of $0 \dots 1$. Neural networks also usually incorporate a bias term to prevent zero-sum inputs skewing and even ruining the learning of the neural network. [4], [6], [24]

Neural networks could also be titled as learning networks. While they could be built with hard-coded values for every distinct key element, the paradigm of biological neural networks has the essential core in the ability to *adapt* and *learn*. Neural networks incorporate this learning by adjusting weights using usually pre-defined learning rate for smoothing or restricting the speed of learning to smaller increments. In a way, the learning rate ensures that the resolution of the teaching process is fine enough for the model to adapt to more finer changes as well. The adjustments are calculated for example from error value. [2], [24]

Neural networks are composed of multiple neurons arranged in distinct layers. The arrangements are subject to design decisions and constitute the topology of any operating neural network. Basic topology in a feed-forward network includes an input layer, at least one hidden layer and an output layer. The number of neurons in input layer correspond to the number of input variables. The number of output neurons in their corresponding layer equal to the number of desired output values. An illustration of a simple neural network topology is depicted in Figure 7. The mappings between the layers and neurons within the layers vary according to the chosen neural network, but the overall idea of multilayer multi-neuron topology with at least layer-wise connectivity is common to all neural networks. [4], [6], [24]

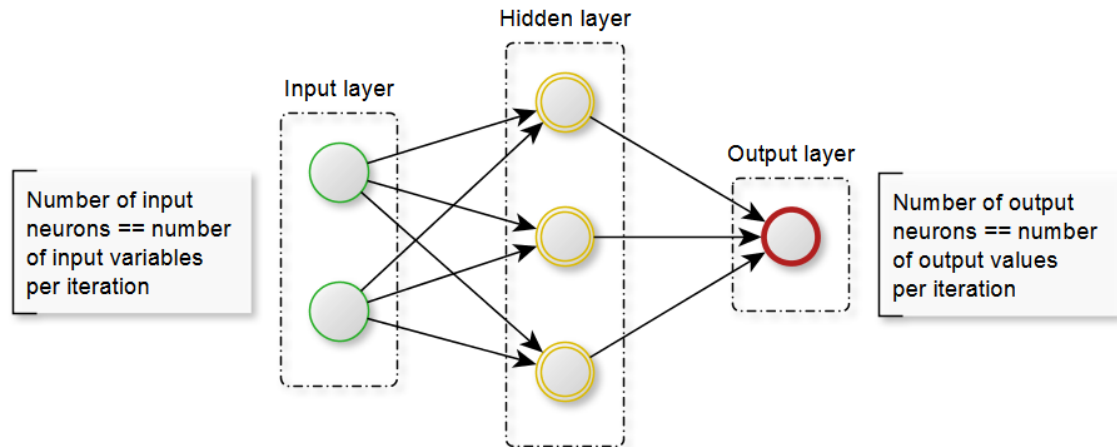


Figure 7. *A simple neural network topology consists of an input, one to several hidden and an output layer.*

An essential element that makes the neural networks non-linear classification and regression models is the non-linear activation function found in most of neural networks. Some neural networks have activation function in every layer's neurons excluding the input layer (e.g. multilayer perceptrons), some incorporate the activation function in only a single layer (e.g. radial basis function networks). These functions transform the input to the ranges of $[0,1]$ or $[-1,1]$ with either hard-limiting the value to extremes, linear ramps or in curved sigmoid or logistic function like fashion. [6], [24]

3.4 Multilayer Perceptrons

While multilayer perceptrons (MLPs) are a complex joining of multiple fully connected single perceptrons to form a neural network topology, the MLP is only a sub-class of neural networks albeit an important one. The basic topology and working principles follow those already introduced in previous sub-chapters. MLPs make use of the back-propagation algorithm to attain knowledge about the dataset for better generalization. To further elaborate on the learning process for the MLPs, the back-propagation algorithm is a two-pass supervised learning weight adjusting error correction method. [4], [24]

The first pass is the forward pass where first every input is weighted and then summed for each hidden neuron in the network. Then the weighted sum of the inputs is passed through a nonlinear activation function, which then forms the weighable input to the subsequent layer all the way to the output layer. A visual representation of the forward pass is depicted in Figure 8 in simplified enough form. The hidden layer act as feature detectors as they map the incoming weighed sums of inputs to new feature space through nonlinear activation function. [4], [24]

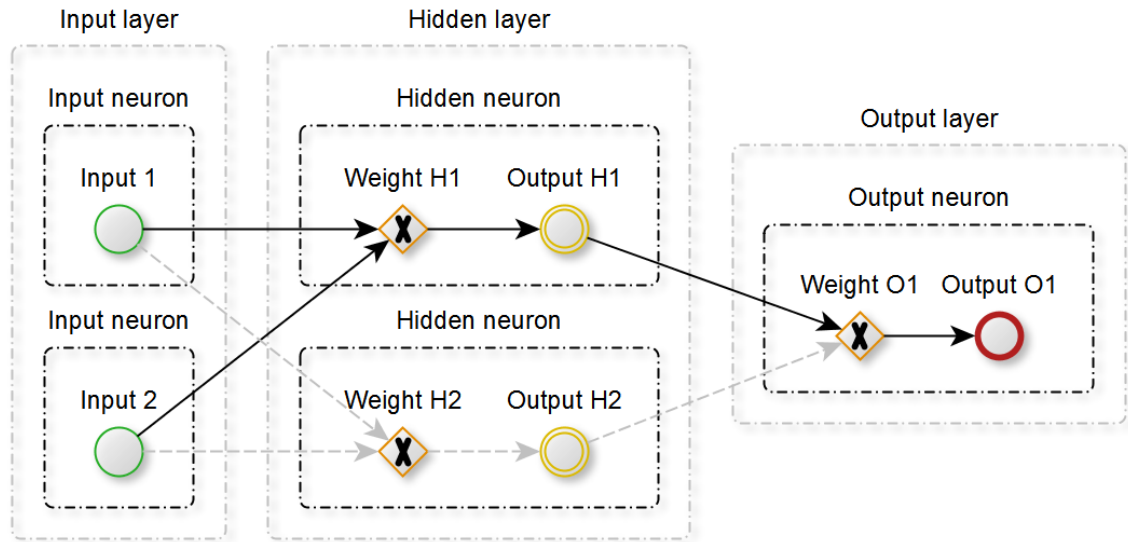


Figure 8. Simplified representation of the forward pass, where preceding layers' outputs are multiplied with neurons' weights and passed on as outputs.

The second pass is called the backward pass and this the phase of learning for the network. While the partial derivative mathematics grow in complexity as layers are added, the basic idea is to find the impact of each neuron's weight to final output error. To open this subject enough to be comprehensive, it would be a requirement to introduce the established and tested mathematical principles, but in the scope of this thesis this level of knowledge is sufficient for the reader. A well-structured and informative step-by-step article can referenced to satisfy the appetite for deeper understanding of the subject [26]. A completed two-pass for a training set of data is called an epoch. [4], [18]

MLPs, like other machine learning methods operating within the sphere of supervised learning, benefits from cross-validation in the training phase. For quick rehearsal, the cross-validation's idea is to slice the training data in to several parts and use one part for testing (or validation, as it is also sometimes called) and then the remaining parts for training. This process is then repeated so that every slice has played the part of the test set.

This subset of neural networks is not a silver bullet though. The MLPs, even though having the reputation of behaving otherwise, sport a risk of getting trapped in a local minimum instead of the global minimum. MLPs also aren't as scalable as some other machine learning models, because the increase in layers also directly affects the amount of required computations per epoch [4]. The training and setting up of the model also takes much more timed when compared to more traditional machine learning concepts introduced in the preceding chapter [27].

3.5 Convolutional Networks

Convolutional networks are a special subset of the MLP paradigm especially concerned with the structure of the network. Where the MLP is fully connected having connections from all previous layer's neurons to subsequent layer's neurons, the convolution network utilizes what is called weight-sharing. This means that each hidden neuron and its weight is shared between a set number of inputs and could also be understood as having only partially or segmentally connected layers. This results in smaller amount of weights versus a fully-connected MLP, which in turn translates to less calculations per epoch. Being a subset of MLPs, the convolutional networks also utilize the back-propagation algorithm. [4], [28]

An example would be a network with six inputs, three hidden neurons and an output neuron. When constructed in the way of a convolution network, the first hidden neuron would sum the inputs from first to fourth, second hidden neuron would sum the inputs from second to fifth and so on. A graphical representation of this is in Figure 9. This is though just only a simplified representation of a single feature extraction without sub-sampling.

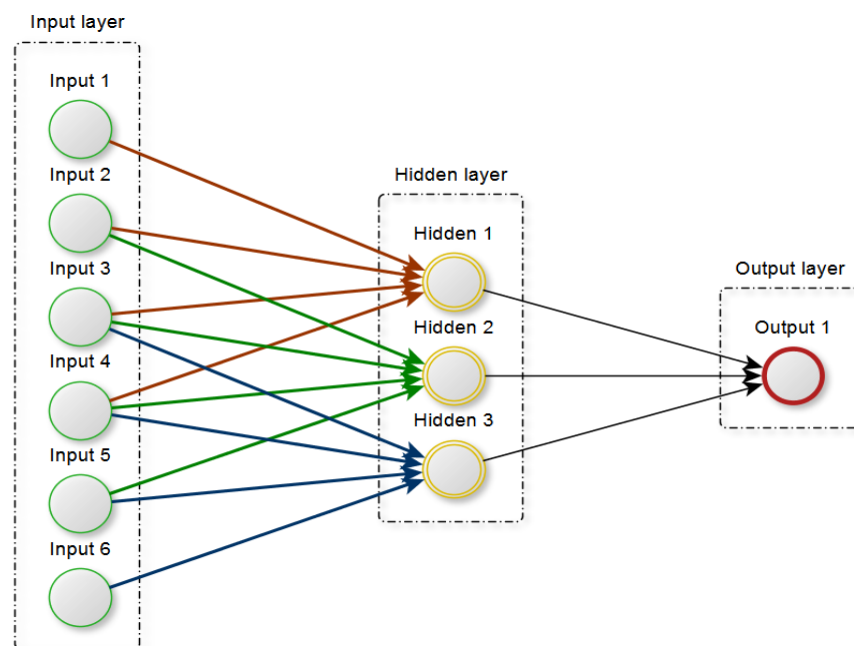


Figure 9. An illustration of a convolutional neural network with three local receptive fields for only one hidden layer.

The reasoning behind this is to create local receptive fields within the neural network for feature extraction. If you take look again at the figure depicting the convolutional network, you can see that the hidden neurons map only a select number of inputs to an output. This way each hidden neuron extracts a local feature out of select inputs. These locally extracted features then constitute a feature map for the set of inputs. Using the figure

again as example, first hidden neuron would describe the feature of inputs with red outward lines, second neuron would describe the features for green line neurons and the third the blue lines neurons'. This kind of approach is extremely useful in pattern recognition from images or audio spectra, where there are multitudes of input variables. [4], [28]

After the inputs have gone through the process feature mapping or in other words, convolution summing, the next phase is to subsample the feature maps. This means the shrinking of the size of the feature maps by calculating for example the averages of pixel pairs. This is also referred to as squashing. The process of convolution and subsampling is then repeated until the desired single feature output is achieved. While the process is a tedious one to understand using verbose measures, a simple imagined process of convolution network extracting feature maps and subsampling them from an input grid of 64 neurons for final output six possibilities of two-digit values is depicted in Figure 10. This kind of convolutional network would have to have a hidden layer of neurons for each phase of convolution and subsampling. [4], [28]

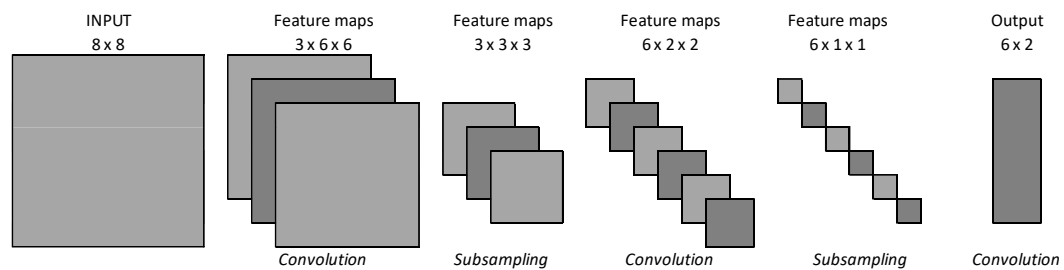


Figure 10. Depiction of convolutional network's convolution and subsampling process for achieving output.

Variations and ensembles of convolution neural networks have performed extremely well in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) during the last few years for image classification tasks [29].

3.6 Radial-Basis Function Networks

The operative premises on which the radial-basis function networks (RBFN) are built upon is the differentiating factor from MLPs utilizing the back-propagation algorithm. The RBFN views the neural networking as a solution to a curve fitting approximation problem, where the classification is acquired by measuring the radial distances to radial basis function neurons within the hidden layer. Where MLPs work hard to define correct feature spaces and their relations, the RBFNs view the fitting as finding the best describing surface of circles, spheres or even hyperspheres within multi-dimensional feature space. [4], [30], [31]

Noteworthy is though that both the MLP and the RBFN are nonlinear layered feedforward networks and are used as data describing model creating tools, the activation functions in

hidden layers are different. The MLP has a non-linear activation function in every layer except the input one, but the RBFN has radial basis functions as the activation functions in its hidden layer's neurons and linear ones in the output layer. This difference in hidden layers results in RBFN modelling clusters for the data rather than MLP-like class boundaries. In simplified terms, the RBFN shares similarities with k -nearest neighbour clustering techniques described in previous chapter. [4], [30], [31]

The topology also is different from the MLPs. At the most basic form, the RBFN has only three layers, meaning also that the number of hidden layers is strictly limited contrary to the MLP concept. First of the layers is the input layer constituted of source nodes. The second layer is the hidden one with its units each providing a radial-basis function, thus making the hidden layer a set of functions providing normalized input pattern's distance to the centre of a single radial basis function. Output layer then maps the inputs from the hidden layer linearly, as opposed to nonlinear output activation functions used with MLPs. The output layer is also the only layer with trainable weights. A simplified illustration is depicted in Figure 11. [4], [30], [31]

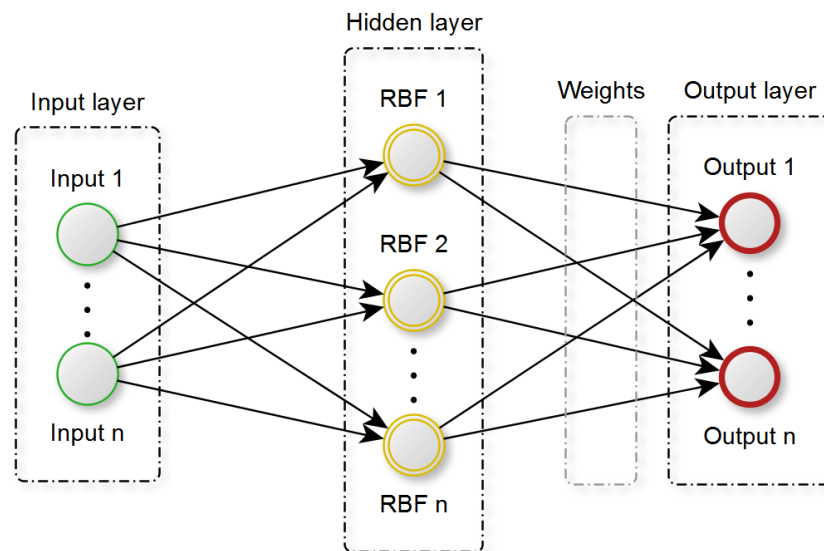


Figure 11. The topology of a radial basis function network with radial basis functions as hidden layer's neurons.

The fit of the model is measured in statistical terms and generalization is equivalent to defining the surface best interpolating between the training and operative data. The original idea proposed in 1960's was that the pattern-classification problem could be linearly easily separable if the problem was cast nonlinearly to a high-dimensional space, which essentially translates to easier classification. In some cases, though, the nonlinear mapping is enough, that is, the increase in dimensionality doesn't add much to solving the problem. RBFN's learning process draws some parallelism from that of MLP's, but is still essentially different. The hidden layer's nodes, regardless of the chosen theoretical background for the network, adjust the weights nonlinearly while the output nodes do the

same linearly. This means that the layer's optimize at different time-scales, as opposed to MLP network aiming to optimize the weights of every neuron during an epoch, which meant a completed forward- and backward-pass for an input dataset. [4], [30]

Radial-basis function networks have been used in a variety of applications, some of these being predicting medical examination results and comparing the predictions to traditional linear and tree models [32], energy production predicting in wind power systems [33] and the identification of black plastic as a part of a composite model using also principal component analysis and linear discriminant analysis [34].

Radial-basis function networks are not without problems. The RBFN learning process is one of reconstructing a multidimensional surface from a sparse dataset. Too large datasets tend to employ massive amount of data with small amount of concrete information relating to the problem and its desired solution – this relates to the existence of distinct outputs for inputs. There might also not be enough information within the dataset overall which can lead to overfitting. Too high noise levels also can degrade the model to provide outputs outside the desired range. In the context of RBFN's hypersurface construction, these three factors affect the stability of the model, but are actually serious source information related problems for every machine learning model that usually cannot be bested with mathematical mingling, but require different approaches to gathering the source data itself. [4], [24]

3.7 Self-Organizing Maps

Competitive learning is a subclass of unsupervised and a category in which the self-organizing maps belong to. In the competitive learning model the output neurons compete in a winner-takes-it-all fashion, where the winning neuron is the one providing the output for a set of source data. The way this is achieved is in resemblance to how Anti-Hebbian learning is achieved in adaptive PCA network – through lateral connections between the output neurons. The architecture for a self-organizing map is a lattice or a grid of commonly one (array) or two (surface) dimensions and the model constructs a topological map of input patterns, where the ordering and the location of output neurons indicate the features in the input datasets. Self-organizing maps are a nonlinear parallel to the linear output PCA and takes much of its inspiration from neurobiological principles of competition, cooperation and self-amplification. The topological ordering and location of output neurons relates to how the brain is designed. Neurons responsible for similar activities are grouped near other similar neurons for shorter synaptic connections, effectively rendering distinct parts of the brain responsible for different main function (i.e. speech, hearing, visual and motor functions). Therefore, the location of the neuron in self-organizing is used to dictate the distinct feature it corresponds to. [4], [10], [35]

Self-organizing map aims at transforming the incoming signal to one- or two-dimensional space irrespective of the input space's dimensionality in adaptive and ordering fashion.

A simplified visual representation of this input-output mapping to two-dimensional surface is in Figure 12, where every input is routed to every output. [4], [35]

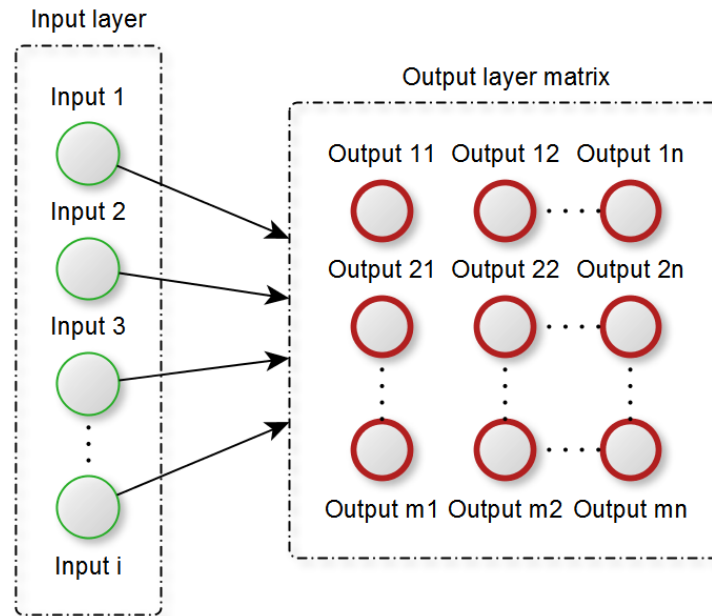


Figure 12. Mapping of inputs to two-dimensional layer of outputs omitting the lateral connections between the outputs.

Like in all adaptive neural networks, the inputs are weighted and summed for output neurons. The competition part of the model happens during the weighing process, and the neuron with highest sum wins the competition for that single epoch of input data. The winner neuron then becomes the topological centre for that input pattern or feature. This is then followed by the cooperation phase, where the laterally closest neurons to the featural topological centre are amplified more than those farther away. Using the above figure as an analogy, if neuron called “Output 11” was deemed as the winning neuron, then the closest neurons would be amplified for that pattern more strongly than those nearing the other corner of the output matrix. In the adaptive phase of the learning process the Hebbian one-way learning resulting in fully saturated connections is overridden with a decay of forgetting factor. [4], [10]

After modelling is finished, the feature map provided by the self-organizing map provides statistical insights about the input data. The model works well as an approximator of the input space from continuous real number space to discrete output space. The topological ordering also is a formidable classifier of the data and feature selector as well. The model is simple to implement but at the same time mathematically difficult to analyse. [4], [10], [35]

3.8 Summary

This chapter discussed the theme of deep learning networks, DLNs. First, a brief introduction to the idea behind deep learning networks and history of the paradigm was given. After the historical review the core concept of neural network learning, the back-propagation algorithm was also discussed. Then the key elements of neural networking were introduced, including the neurodynamic components related to the topology of neural network (i.e. neurons and weights) accompanied with modelling-related parameters such as the learning rate.

Then the chapter introduced four distinct DLN methods. First of these was the MLP, which is a layered combination of multiple linear perceptrons. The MLP is one of the most fundamental concepts within the DLN paradigm and can handle i.e. regression and classification. Second was the convolutional network, which is the dominant go-to method in image recognition. Second to last of the methods was the radial-basis function network, which shares similarities with the nearest-neighbour modelling method but in a DLN context. Last of the DLN methods addressed was the self-organizing maps or the SOM, which is a neural network clustering application.

4. MACHINE LEARNING METHOD COMPARISON

After reviewing some of the most used concepts in the fields of traditional machine learning and deep learning, the comparison is a natural progression towards conclusions on methods' performances and viabilities. This chapter focuses therefore solely on comparing the reviewed methods within appropriate machine learning tasks using Python and its associated or user-developed machine learning packages as the operative research framework.

The machine learning method comparison is done for two main tasks, clustering and regression. As the title of the thesis suggests, this is done using data accumulated from the Finnish agriculture, and more accurately from the Finnish dairy farms. The datasets used in the research were originally comprised of several tables relating to cattle and single bovine milk production information, bovine health records, insemination information and overall farm statistics. Two distinct datasets were then constructed from the source data for both clustering and regression tasks. The compilation of datasets was in both cases then followed by data preparation, after which the machine learning methods were applied to the datasets.

The chapter begins with comparison of four clustering methods, namely Hierarchical Clustering, k -Means, Self-Organizing Maps and BIRCH. First the preparation of datasets is addressed, after which the methods' Python-wise implementations are introduced. Then the clustering methods are compared to each other. Then the chapter progresses to address select regression methods in analogous manner. The regression methods addressed are Ordinary Least Squares, Decision Tree Regression, Multilayer Perceptron and XGBoost. First the dataset formation is addressed, which is followed with introductions of the Python-wise regression method implementations and lastly the methods are compared with each other.

4.1 Clustering

The first machine learning goal set by the employer of the thesis was to find out about the differences between farms with differing configurations. Core research question for this machine learning task can be divided in to two parts, which are the following:

3. *“Are there notable differences related to production between farms and their configurations?”*
4. *“Are there notable differences related to cattle health between farms and their configurations?”*

The configurations effectively combinations of several types of bovine environment and milk production related factors, such as housing, milking machine and feeding type. The configuration variables are defined as follows:

- Housing type with three possible types
- Milking machine type with nine possible types
- Cow traffic type with two possible types
- Feeding type with three possible types
- Milking cows going out with five possible types.

It is to be noted that these variables can't properly be handled as dependent variables. While the dataset could provide a possibility for supervised classification by attaching an identification label to each possible combination of dependent variables, the use of classification is effectively rendered obsolete by the closeness between the number of possible combinations (810) and a relatively small number of input rows (834). It was therefore deemed better to use unsupervised clustering as the operating paradigm for this dataset.

The clustering was decided to be done with production and health related records, which comprised the input dataset. The input dataset contains records about the following subjects:

- Bovine basic information (i.e. age)
- Cattle milk production (i.e. overall production, nutrition metrics, mean production per bovine)
- Bovine milk production (i.e. lifetime milk production)
- Bovine illness treatment records
- Bovine insemination records.

The plan for finding differences for configurations across different methods is to first train the clustering algorithms with input dataset only, which are indexed by farm identification numbers called *Idkar*. The training is done with two distinct input datasets:

- With production variables
- With health variables

After clustering, the configuration datasets are sorted for each method separately according to produced clusters and the configurations are statistically examined. Most interesting statistic is the mode of the five configuration types, as the types are categorical features having discrete values.

4.1.1 Preparing the Datasets

First step in the process of clustering algorithm comparison is the preparation of the datasets, two different input datasets and one for linking the farm configurations to the farm identifiers to be separated from one combining dataset. Originally the strategy was to use only one dataset, but as the research progressed it became evident that the separation was necessary. Main indicator was the inconsistent clustering results for the health treatment variables only with using clustering configurations optimized for the whole dataset.

The datasets were to be formed out of farms that are eligible for research purposes and bovines that were alive during the research. The first thing in the data preparation was to gather required valid records and fields to a form that's usable by the algorithms in one CSV-file containing all the records and columns. The required data was scattered amongst eight database tables. To gather the data, a single combining SQL-query was written. The SQL-query used in the selection of current data is given in Appendix A for reproducibility. Retrievable records were first filtered by either bovine validity or cattle validity accordingly. After validity checks, the cattle-related production variables were joined with cattle-wise calculations of bovine production and health record means. The original idea was to use means of production variables combined with sums of health-related variables, but the usage of means in both cases seemed more aligned with overall dataset. The number of rows in the combined dataset containing every record eventually totalled to 834 records with 30 columns.

The data was then visualized to explore the possibility of existing outliers before making any dataset splits. As the data is multidimensional, histograms were plotted on some of the dimensions. While all the features could've been plotted, a selection of some features already provided enough information about the existence of a single outlier farm. From Figure 13 it is evident, that there exists at least one record that can be assessed as an outlier. After making sure that these notably higher values originate from the same source, the record deemed as an outlier was then dropped from the dataset. While the algorithms could be robust enough to allow for outliers, the cluster number optimization techniques presented later in this chapter were notably affected by the outlier by allocating the outlier to one, rather small, cluster distinctly dissimilar from other larger clusters.

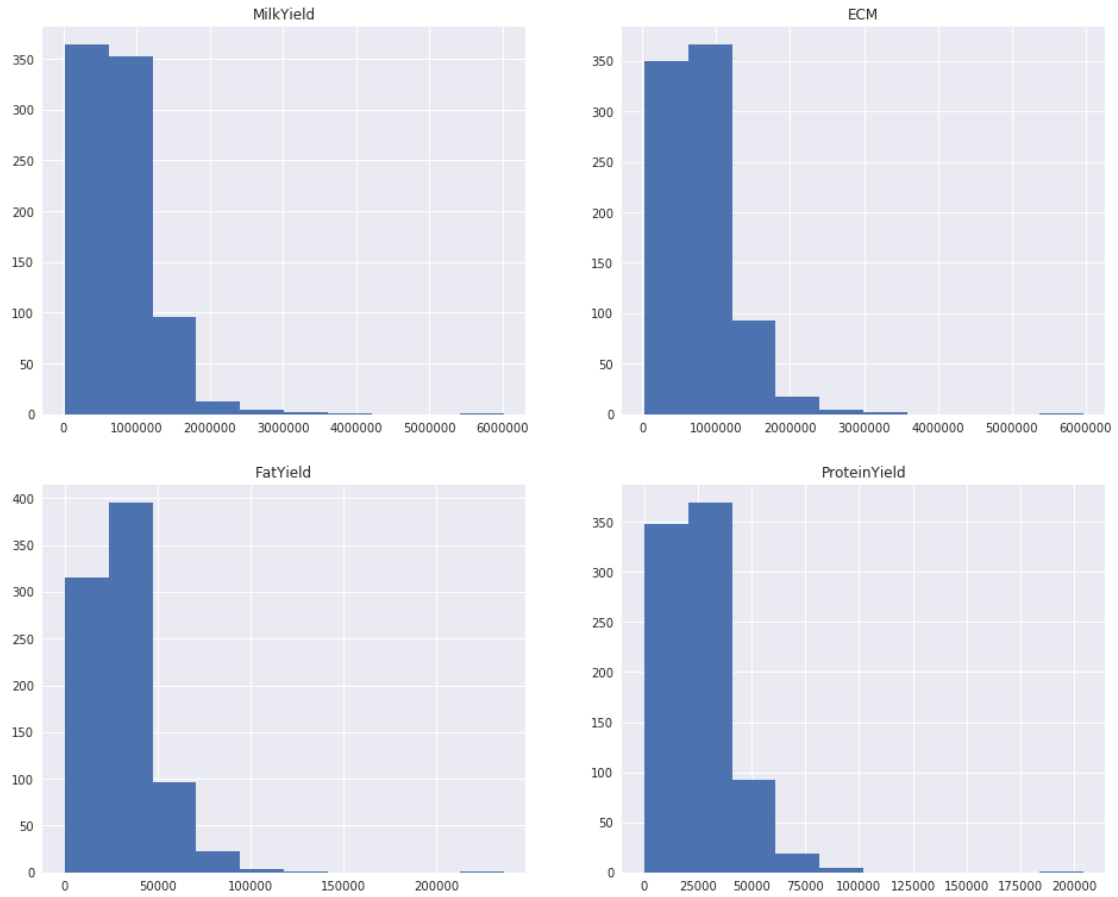


Figure 13. Histograms of select dataset's dimensions reveal an outlier.

Last step in the process of data preparation after outlier elimination was to divide the data to two input datasets and a configuration dataset. To underline the fact, the preparation of distinct input datasets is not for supervised training. The reason for this split is to produce the clusters by training the algorithms without influencing the clustering process with configuration related information. This way the clusters are formed from the input data only and possible distinctions in the configuration rise from production and health related clustering alone.

The input datasets were thus formed by omitting the configuration related columns from the dataset. An overview of the form of the tables and variable value magnitudes of the input datasets are given in Table 1 for production variables and Table 2 for health treatment variables. Both tables are given as transposes of the original dataset for representation purposes. That means, that the columns are on the left side of the table and the row indexes at the top of the table. Production related variables total to 16 variables. These variables were pre-calculated in the database's tables. Health treatment related variables total to seven variables. These were formed with the SQL-query by counting all the treatment occurrences for each bovine and then taking the cattle's bovines' mean for different treatment categories.

Idkar	100001	100015	100074	100228	100329	100987
MilkYield	1.22e+06	1.51e+06	507357.00	1.47e+06	837442.00	679042.00
MilkYieldMean	1.03e+04	1.29e+04	9396.00	1.13e+04	10709.00	9701.00
ECM	1.26e+06	1.47e+06	544967.00	1.54e+06	838693.00	702225.00
ECMMean	1.06e+04	1.26e+04	10092.00	1.18e+04	10725.00	10032.00
FatYield	5.12e+04	5.74e+04	20496.00	6.23e+04	33565.00	28509.00
FatMean	4.33e+02	4.92e+02	380.00	4.77e+02	429.00	407.00
ProteinYield	4.23e+04	5.09e+04	16573.00	5.35e+04	28598.00	23997.00
ProteinMean	3.58e+02	4.36e+02	307.00	4.10e+02	366.00	343.00
FeedingDays	4.33e+04	4.28e+04	19760.00	4.78e+04	28631.00	25603.00
SomCellWMean	1.88e+02	1.33e+02	171.00	2.49e+02	190.00	319.00
CowMean	1.18e+02	1.17e+02	54.00	1.30e+02	78.20	70.00
UnreliableYield	0.00e+00	0.00e+00	0.00	0.00e+00	0.00	0.00
Urea	2.00e+01	2.40e+01	28.00	3.20e+01	24.00	32.00
AvgBovineLifetimeMilkYield	2.59e+04	2.16e+04	21041.13	2.23e+04	18486.51	16552.30
AvgBovineAge	5.18e+00	4.37e+00	4.83	4.47e+00	4.55	4.25
InsemSuccessFactor	3.94e-01	5.19e-01	0.45	5.41e-01	0.62	0.43

Table 1. A transpose of the production variables' input dataset depicting variable names and value magnitudes.

Idkar	100001	100015	100074	100228	100329	100987
TreatFertilityAvg	0.76	1.15e+00	0.89	5.67e-01	0.94	0.08
TreatMetabolismAvg	0.06	4.92e-02	0.00	1.67e-02	0.24	0.03
TreatDigestionAvg	0.03	2.46e-02	0.00	0.00e+00	0.01	0.00
TreatUdderAvg	0.18	3.52e-01	0.36	1.17e-01	0.37	0.07
TreatHoofAvg	0.09	5.74e-02	0.02	8.33e-03	0.45	0.00
TreatRespiratoryAvg	0.00	0.00e+00	0.00	0.00e+00	0.00	0.00
TreatCalfAvg	0.00	8.20e-03	0.00	8.33e-03	0.00	0.02

Table 2. A transpose of the health treatment variables' input dataset depicting variable names and variable magnitudes.

The configuration dataset was then formed by omitting all the other variables or columns than the farm configuration related variables from the original combining dataset. An overview of the configuration dataset is given in Table 3.

	IQFeature1	IQFeature2	IQFeature3	IQFeature4	IQFeature5
Idkar					
100001	3.0	5.0	1.0	3.0	2.0
100015	2.0	1.0	1.0	1.0	5.0
100074	2.0	1.0	1.0	1.0	2.0
100228	2.0	1.0	1.0	3.0	5.0
100329	2.0	1.0	1.0	3.0	2.0
100987	2.0	3.0	1.0	2.0	1.0

Table 3. Configuration dataset contains information about farms' configurations.

An *IQFeature* corresponds to a distinct configuration type and the values correspond to the category of within the type, latter of which are pre-defined categorical values in the source system. The configurations and their values in plain language are the following:

- IQFeature1: Housing type
 - 1: Tied up
 - 2: Loose
 - 3: Other
- IQFeature2: Milking machine type
 - 1: Automatic milking
 - 2: Other milking parlour
 - 3: Automatic milking and parlour
 - 4: Rotary parlour
 - 5: Side-by-side parlour
 - 6: Tandem parlour
 - 7: Herringbone parlour
 - 8: Pipeline milking
 - 9: Bucket milking
- IQFeature3: Cow traffic type
 - 1: Free
 - 2: Directed
- IQFeature4: Feeding type
 - 1: Individual normal feeding
 - 2: Flat rate feeding
 - 3: TMR feeding
 - 4: TMR feeding with concentrate boxes
 - 9: Other
- IQFeature5: How the milking cows get outdoors
 - 1: Grazing with walks in the winter

- 2: Grazing with indoors all winter
- 3: Walks both summer and winter
- 4: Walks in the summer, in all winter
- 5: Indoors all year round

While the data in the configuration dataset is digits only, the above listing is adequate for reviewing the configuration types' descriptions. In other words, the farm configuration variables are categorical features, but are not used in training the clustering methods. The configuration types are matched to farms only post-clustering.

4.1.2 Optimal Number of Clusters

Next step after the preparation stage is to define the optimal number of clusters separately for the production and health treatment datasets, as some of the clustering algorithms require user to input the number of clusters manually in Python. Two cluster number estimation techniques were used for this, mainly to gain certainty about the result. These were the dendrogram and what is known as the Elbow Method.

The dendrogram has already been introduced in Chapter 2 under Hierarchical Clustering, but to recapitulate, a dendrogram illustrates the closeness of data points by drawing U-shaped links between them. The distance of the link's ceiling from the x-axis depicts the dissimilarity between the data points or groups of data points with greater distance translating to greater dissimilarity. While the build-up of the dendrogram is a bottom-to-top approach, the cluster formation progresses from top to bottom. The cluster with highest dissimilarity is always the one to split, meaning that two clusters would equal to green and red cluster, three to two red and a green and four to two green and two red clusters. Because there were two datasets, two dendrograms were formed for both. The dendrogram for the production variable dataset is shown in Figure 14 and the dendrogram for the health treatment dataset in Figure 15.

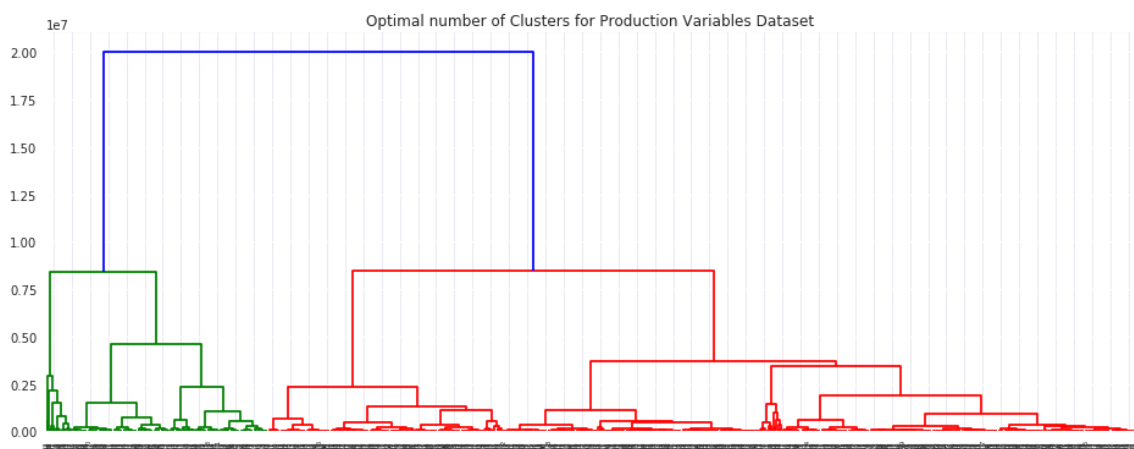


Figure 14. Dendrogram produced from the production variables dataset.

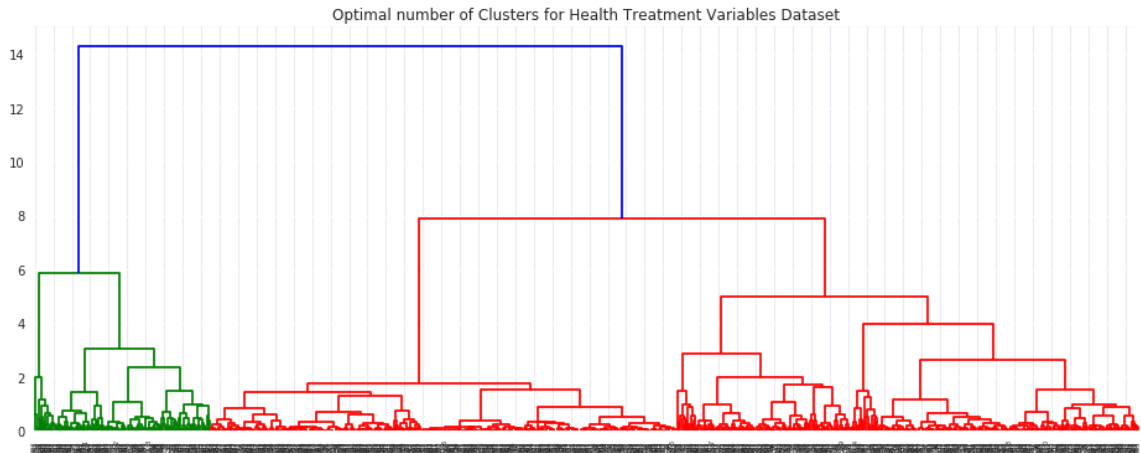


Figure 15. Dendrogram produced from the health treatment variables dataset.

The Elbow Method is quite different approach at finding the optimal number of clusters to the dendrogram. The Elbow Method makes use of k -Means algorithm by training the data with multiple k -values and using the calculated within-cluster sums of squares as a gauge for complexity change with varying cluster sizes. Optimal cluster number is the k -value that acts as a pivot point for noticeable change in the decrease of within-cluster sums of squares (WCSS). When applied to the datasets, the Elbow Method indicates that the number of clusters should be between two and four for the production variables dataset (Figure 16) and health treatment variables (Figure 17).

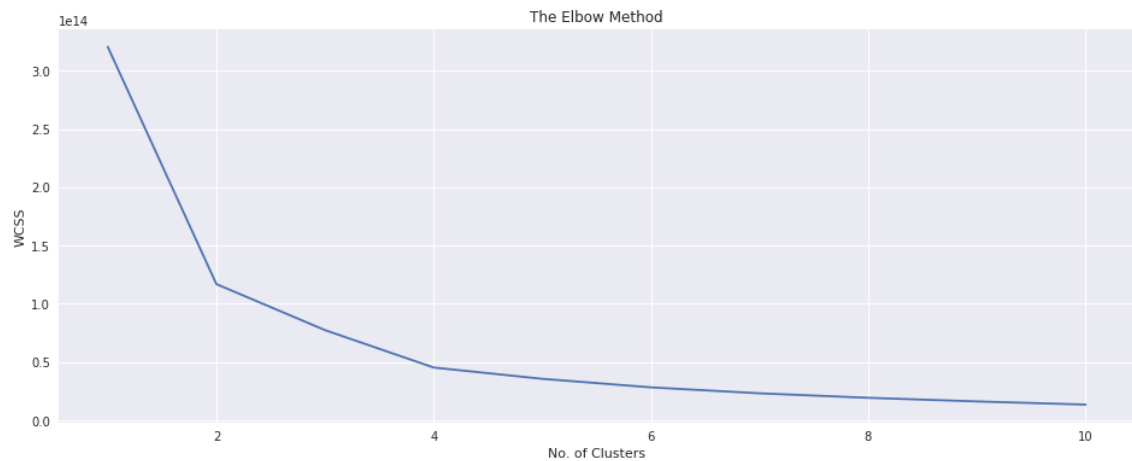


Figure 16. Elbow Method WCSS progression for the production variables dataset.

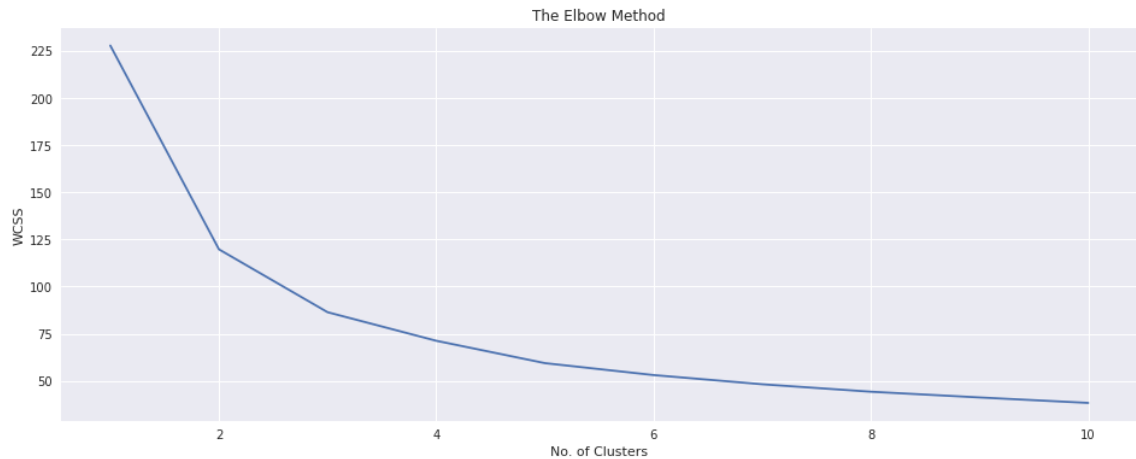


Figure 17. Elbow Method WCSS progression for the health variables dataset.

Both techniques seem to point towards an optimal number of two clusters for both datasets. While the number is instructive, clustering could also be done by three or even four clusters in both cases. The decision was made to use three clusters with production variables dataset for optimization of equal cluster sizes over similarity of cluster dissimilarity. Two clusters were chosen then for health treatment variables for optimizing the equality of clusters' dissimilarity contrary to the other dataset.

4.1.3 Hierarchical Clustering

First clustering technique introduced to the comparison of clustering methods was Hierarchical Clustering (HC). The package used for computation *AgglomerativeClustering*-class found in the *sklearn.cluster*-module. The clustering object was initialized to use Euclidean distance as the metric for linkage computation. Other possible metrics are for example Manhattan and cosine distances. The minimization criterion was initialized to minimize the sum of squared differences within clusters akin to k-Means but with hierarchical bottom-to-top approach. Other possible minimization criteria are the maximum distance minimization and average distance minimization, all of which have the goal of optimizing the clusters in their own way. [14]

The modes of configuration variables across clusters formed with HC are depicted in the following tables with two input dataset settings – production variables only in Table 4 and health variables only in Table 5.

	IQFeature1	IQFeature2	IQFeature3	IQFeature4	IQFeature5
0	2	1	1	3	5
1	2	1	1	1	5
2	2	1	1	1	2

Table 4. Modes of configuration values using production variables as input dataset with HC.

	IQFeature1	IQFeature2	IQFeature3	IQFeature4	IQFeature5
0	2	1	1	1	5
1	2	1	1	1	5

Table 5. Modes of configuration values using health variables as input dataset with HC.

The resulted cluster sizes are depicted in Table 6. The sizes seem to visually correspond to the dendrograms and their depicted cluster size ratios given in previous portion regarding the preparation of the datasets.

	Production variables	Health variables
Cluster 1	170	700
Cluster 2	484	133
Cluster 3	179	-

Table 6. Cluster sizes of HC.

Initially it seems that HC gives no surprises with dendrogram already present in the toolbox – the ratios of cluster sizes match well with the dendrograms’ figures. What is interesting though is the fact that for production variable dataset there seemed to be a single key configuration element differentiating the clusters from each other. Health variables dataset however does not seem to have any configuration related differentiating factor between the clusters. The statistical analysis of clusters’ values is presented at the end of the clustering section, which then gives insight into the differentiation between health variable dataset’s clusters.

4.1.4 *k*-Means

k-Means was the next clustering method to be taken under scrutiny. The operating principle of this method has already been demonstrated during the data preparation with the Elbow Method, which illustrates efficiently the effect of varying the *k*. As a clustering method, it has orthogonally opposite approach to HC by converging from top-to-bottom. The package is called *KMeans* and is found in module *sklearn.cluster*. Apart from defining the number of clusters used, the algorithm was run with default settings. Most important of these unchanged initialization parameters is the centroid initialization function, that can be set to either smart or random initialization. Smart initialization performs preliminary clustering to find somewhat optimal initial for sped up convergence while random initialization chooses a defined number of random data points as initial cluster centroids. [14]

Like the previous algorithm, the k -Means was also trained with the production variables dataset and health treatment variables dataset. The configuration modes are listed respectively in Table 7 and Table 8. The production variables dataset clusters seem to differ from HC at first glance, but the true difference might lie only in different indexing of the clusters. A look at the cluster sizes and later into statistical values of clusters' variables together either validate or disprove the hypothesis, but the initial intuition is that the clusters are after all similar. Health variable dataset's clusters seem as well similar to HC when considering the configuration types only.

	IQFeature1	IQFeature2	IQFeature3	IQFeature4	IQFeature5
0	2	1	1	1	2
1	2	1	1	3	5
2	2	1	1	1	5

Table 7. Modes of configuration values using production variables as the input dataset with k -Means clustering.

	IQFeature1	IQFeature2	IQFeature3	IQFeature4	IQFeature5
0	2	1	1	1	5
1	2	1	1	1	5

Table 8. Modes of configuration values using health variables as the input dataset with k -Means clustering.

The sizes of the k -Means clustering method's clusters are shown in the Table 9. The cluster sizes are not identical and therefore do not form links straight away between HC's clusters, but the ratios of cluster sizes align with nevertheless.

	Production variables	Health variables
Cluster 1	482	653
Cluster 2	138	180
Cluster 3	213	-

Table 9. Cluster sizes of k -Means clustering.

4.1.5 Self-Organizing Maps

After using two distance-based clustering methods, a DLN clustering model is more than fitting addition to the group of comparable clustering algorithms. SOM accomplishes the task of clustering by training a lattice of output neurons specializing to distinct input patterns. SOM is essentially a winner-takes-it-all algorithm, where the neuron with highest

output is the only output for the input. While being the winner, it also is empowered to perform even better for consecutive similar inputs elevating the specialization even further. A more complete overview of the intuition behind SOM is given in Chapter 3. The package used for training the SOM model is called *SOMPY* and is developed by a group of individual users, meaning it is not part any out-of-the-box basic python libraries such as *sklearn*. The initialization of the SOM module takes essentially a single required input parameter which is the size of the lattice or the output neuron map. The size was decided to be 15x15 neurons for adequate output lattice resolution, but it could've been also bigger at the cost of increased computation time.

Because SOM is essentially all about a two-dimensional lattice of output neurons, the output lattices depicting the output neuron cluster boundaries called hitmaps were produced for both datasets, the production variable dataset (Figure 18) and the health treatment variable dataset (Figure 19). The hitmaps can be used to fetch dataset's rows' clusters by first feeding the data row-by-row to the SOM, marking the winner neurons up and then checking the cluster to which each winner neuron belongs to.

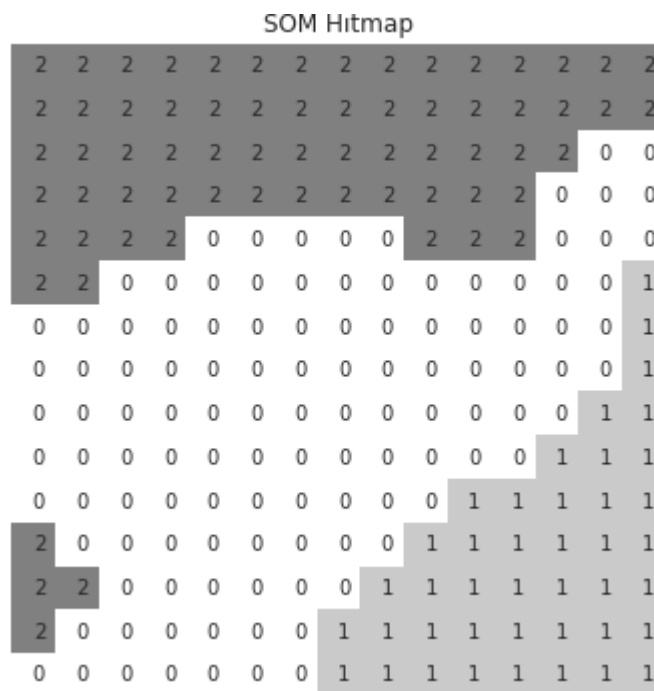


Figure 18. SOM output layer with cluster boundaries using production variables dataset.

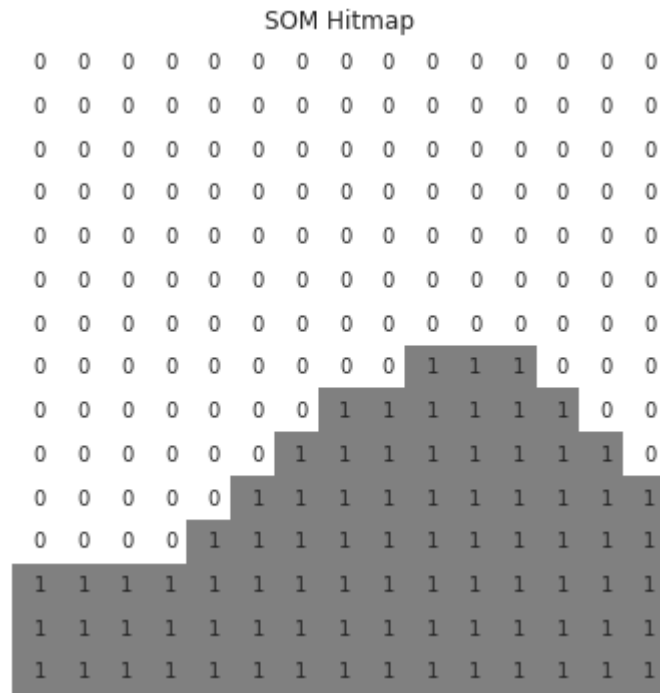


Figure 19. SOM output layer with cluster boundaries using health treatment variables dataset.

It is also noteworthy to mention that the ratios of output lattices cluster sizes correspond well to actual cluster sizes. For production variables dataset, the corresponding cluster sizes using the hitmap ratios only would be 572 rows for cluster at index 0, 155 rows for cluster at index 1 and 251 rows for cluster at index 2. The estimated row numbers for the health variable dataset's clusters would be 525 and 307 rows respectively. While these are not the real row number, they seem to give an estimation of the partitioning ratios for the clusters.

The calculations of clusters' configuration type modes reveal no significant secrets. Again, the configuration types experiencing differentiation are the last two and the differentiation is only with one variable against the other clusters for the production variable dataset (Table 10). The health variable dataset's clusters' modes are also similar to already used algorithms' results (Table 11).

	IQFeature1	IQFeature2	IQFeature3	IQFeature4	IQFeature5
0	2	1	1	1	5
1	2	1	1	3	5
2	2	1	1	1	2

Table 10. Modes of configuration values using production variables as the input dataset with SOM clustering.

	IQFeature1	IQFeature2	IQFeature3	IQFeature4	IQFeature5
0	2	1	1	1	5
1	2	1	1	1	5

Table 11. Modes of configuration values using health treatment variables as the input dataset with SOM clustering.

The estimation of the clusters sizes wasn't quite on the spot with SOM output layer visualizations, but still close enough when looking at real row counts for clusters using SOM. If something is worth mentioning, it is the fact that SOM makes yet the most even distribution of rows between clusters when using health treatment variable dataset. The correct cluster sizes are given in Table 12.

	Production variables	Health variables
Cluster 1	451	563
Cluster 2	139	270
Cluster 3	243	-

Table 12. Cluster sizes for Self-Organizing Maps.

4.1.6 BIRCH

The last one of the clustering methods was a clustering tree, which seemed like a good addition to the comparison but also was a bit of a wild card – it wasn't found in the literature used for the theoretical portion of this thesis but in the documentation of Python's *sklearn*-module during the concrete research. More specifically, the "Balanced Iterative Reducing and Clustering using Hierarchies"-algorithm known with the acronym BIRCH is a clustering feature tree. In essence it is a hierarchical clustering algorithm in a branching tree form with the main advantage postulated at providing higher performance with larger datasets [36]. The Python package used is named *Birch* and is found in the library *sklearn.cluster*. The documentation for the algorithm also states, that when the number of clusters is defined by the user, the algorithm uses agglomerative clustering to fit the model further validating the point of the algorithm being essentially a hierarchical clustering algorithm, albeit a tree one. If no cluster number is defined, the algorithm performs more like a tree, calculating the clusters purely using branching factor and threshold [14]. The algorithm was eventually initialized with a set number of clusters. The production variable clusters were identical with and without a set number of clusters, and the health treatment variable clusters were even worse divided without than with a set number of clusters.

Like with all the other clustering methods, first part was to calculate the modes of configuration values across generated clusters. The production variable dataset's clusters

(Table 13) are again similar to previous algorithms' clusters and the same can be said for the health treatment variable dataset (Table 14).

	IQFeature1	IQFeature2	IQFeature3	IQFeature4	IQFeature5
0	2	1	1	3	5
1	2	1	1	1	5
2	2	1	1	1	2

Table 13. Modes of configuration values using production variables as the input dataset with BIRCH clustering.

	IQFeature1	IQFeature2	IQFeature3	IQFeature4	IQFeature5
0	2	1	1	1	5
1	2	1	1	1	5

Table 14. Modes of configuration values using health treatment variables as the input dataset with BIRCH clustering.

The cluster sizes however show some striking results, as depicted in Table 15. First observation relates to the production variable clusters, which are identical in respect to sizes and the ordering of the clusters with those of HC. The underlying reason is most probably in the use of agglomerative clustering, i.e. hierarchical clustering when the number of clusters is defined. The other observation is the most extreme unbalanced allocation of rows with the health treatment dataset.

	Production variables	Health variables
Cluster 1	170	817
Cluster 2	484	16
Cluster 3	179	-

Table 15. Cluster sizes for BIRCH.

4.1.7 Comparison of Clustering Methods

After examining the methods individually, it is finally time to perform comparison of the methods in more detail. At first it is in place to provide the overall distribution of configuration dataset's values. The modes of the configuration type values are presented in Table 16. These are essentially the dominant values across the whole configuration dataset.

	IQFeature1	IQFeature2	IQFeature3	IQFeature4	IQFeature5
0	2.0	1.0	1.0	1.0	5.0

Table 16. *The modes of configuration type values within the whole configuration dataset.*

The stacked histogram depicting the overall distribution of values is also given in Figure 20. This is to help identify the clusters that differ from the largest group of farms (the production variable dataset) and when there is no configuration related clustering visible (the health treatment dataset).

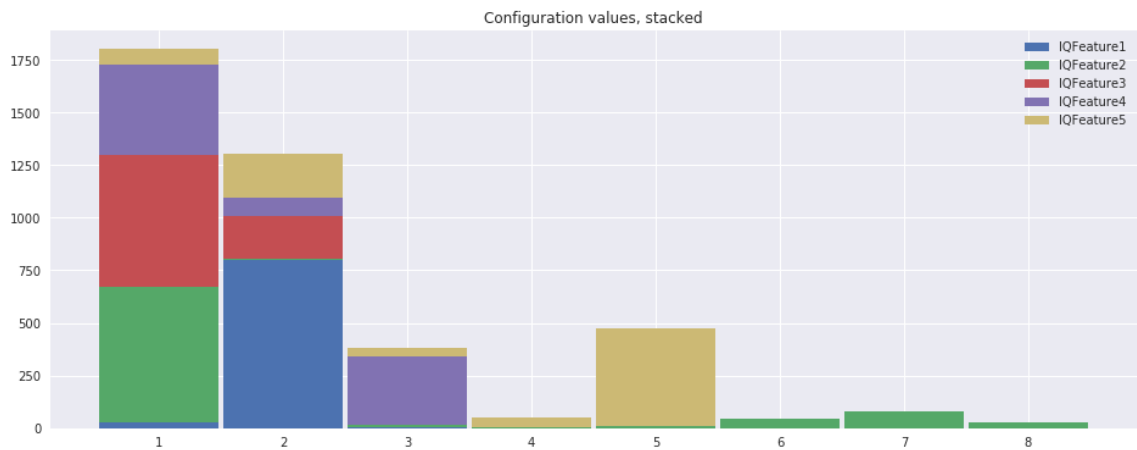


Figure 20. *The distribution of values of configuration types.*

For gaining an easier overview, the clusters' configuration modes of every method were combined to a single list for both production (Table 17) and health treatment (Table 18) variable datasets. The tables contain no additional information in themselves, but are only combinations of multiple smaller tables. While avoiding excessive recapitulation on some already made statements, all the clustering algorithms seem to agree on the separation principles of both datasets; no clustering algorithm stands out as being particularly distinct from the other from viewpoint of configuration value modes. All the methods agree on one differentiating configuration factor for the production dataset and no configuration differentiation for the health treatment dataset. To recapitulate, the tables of modes represent the dominant categorical farm configurations within clusters, which were the point of interest for the employer of the thesis regarding clustering.

		IQFeature1	IQFeature2	IQFeature3	IQFeature4	IQFeature5
BIRCH	0	2	1	1	3	5
	1	2	1	1	1	5
	2	2	1	1	1	2
HC	0	2	1	1	3	5
	1	2	1	1	1	5
	2	2	1	1	1	2
SOM	0	2	1	1	1	2
	1	2	1	1	3	5
	2	2	1	1	1	5
k-Means	0	2	1	1	1	2
	1	2	1	1	3	5
	2	2	1	1	1	5

Table 17. Compilation of configuration type value modes with production variable dataset.

		IQFeature1	IQFeature2	IQFeature3	IQFeature4	IQFeature5
BIRCH	0	2	1	1	1	5
	1	2	1	1	1	5
HC	0	2	1	1	1	5
	1	2	1	1	1	5
SOM	0	2	1	1	1	5
	1	2	1	1	1	5
k-Means	0	2	1	1	1	5
	1	2	1	1	1	5

Table 18. Compilation of configuration type value modes with health treatment variable dataset.

While the sizes of formed clusters have already been addressed, the concrete values for different variables and their distributions amongst clusters are yet to be discussed. With multiple variables against several methods and their clusters, a concise way of illustrating the statistical distributions within clusters in comprehensive manner is desired. This was achieved by calculating each variable's cluster-wise means represented in following two

tables. The values are given in scientific mathematical notation for more concise cells. In both tables the color-coding is as follows:

- A variable's minimum mean is highlighted with red, separately for each method.
- A variable's maximum mean is highlighted with blue, separately for each method.
- Mean values with white backgrounds are in-between the minimum and maximum for a method.

To illustrate this, minimum *Count* for *k*-Means is 138 in cluster at index 1, while maximum is 482 in cluster at index 2; The maximum mean of *ProteinYield* for SOM is around 51000 in cluster at index 1, while the minimum is 11000 in cluster at index 0. The same logic holds for both tables. It also noteworthy to remind, that the following tables depict the variables, that were used for training the clustering algorithms – the configuration types were matched to the corresponding clusters only after the training was finished.

By means of visual cross-comparison, there seems to be two groups of clustering algorithms among production variable dataset (Table 19). When going through the minimum, maximum and middle value columns, BIRCH and HC share identical clusters. The other clusters, SOM and *k*-Means, also seem more related to each other than to either BIRCH or HC. By looking at some variables, the following can be said about all clustering algorithms:

- With *CowMean* it seems, that the algorithms have separated the farms to small, midsize and large farms.
- With *MilkYieldMean* it seems, that there is no significant difference between large and midsize farms, but farms with lowest headcount mean have considerably lower average yield.
- The *InsemSuccessFactor* seems rather indifferent to farm differences – it is essentially the same across the clusters, while clusters with lowest average head count have a few hundredths larger rate than the other clusters.
- The *AvgBovineAge* also seems to a bit higher (around 2½ months) for lower head count farms.

	BIRCH			HC			SOM			k-Means		
	0	1	2	0	1	2	0	1	2	0	1	2
Count	170	484	179	170	484	179	243	139	451	213	138	482
MilkYield	1.4e+06	6.7e+05	3e+05	1.4e+06	6.7e+05	3e+05	3.8e+05	1.5e+06	7.2e+05	3.3e+05	1.5e+06	7.1e+05
MilkYieldMean	1e+04	1e+04	8.2e+03	1e+04	1e+04	8.2e+03	8.3e+03	1e+04	1e+04	8.4e+03	1e+04	1e+04
ECM	1.4e+06	6.6e+05	2.9e+05	1.4e+06	6.6e+05	2.9e+05	3.5e+05	1.5e+06	7e+05	3.2e+05	1.5e+06	7e+05
ECMMean	1e+04	9.9e+03	8.2e+03	1e+04	9.9e+03	8.2e+03	8.2e+03	1e+04	1e+04	8.3e+03	1e+04	1e+04
FatYield	5.6e+04	2.6e+04	1.1e+04	5.6e+04	2.6e+04	1.1e+04	1.3e+04	6e+04	2.7e+04	1.2e+04	5.9e+04	2.7e+04
FatMean	4.1e+02	3.9e+02	3.2e+02	4.1e+02	3.9e+02	3.2e+02	3.2e+02	4.1e+02	4e+02	3.3e+02	4e+02	4e+02
ProteinYield	4.8e+04	2.1e+04	9e+03	4.8e+04	2.1e+04	9e+03	1.1e+04	5.1e+04	2.3e+04	1e+04	5e+04	2.3e+04
ProteinMean	3.5e+02	3.3e+02	2.6e+02	3.5e+02	3.3e+02	2.6e+02	2.6e+02	3.5e+02	3.4e+02	2.7e+02	3.4e+02	3.3e+02
FeedingDays	5.1e+04	2.5e+04	1.3e+04	5.1e+04	2.5e+04	1.3e+04	1.6e+04	5.3e+04	2.6e+04	1.4e+04	5.4e+04	2.6e+04
SomCellWMean	2e+02	1.9e+02	1.8e+02	2e+02	1.9e+02	1.8e+02	1.8e+02	2e+02	1.9e+02	1.8e+02	2e+02	1.9e+02
CowMean	1.4e+02	68	39	1.4e+02	68	39	46	1.5e+02	71	41	1.5e+02	71
UnreliableYield	0.029	0.083	0.21	0.029	0.083	0.21	0.21	0.014	0.067	0.2	0.043	0.073
Urea	28	26	23	28	26	23	24	28	26	23	27	26
AvgBovineLifetimeMilkYield	2.1e+04	2.1e+04	1.9e+04	2.1e+04	2.1e+04	1.9e+04	1.9e+04	2.1e+04	2.1e+04	1.9e+04	2.1e+04	2.1e+04
AvgBovineAge	4.6	4.6	4.8	4.6	4.6	4.8	4.8	4.6	4.6	4.8	4.6	4.6
InsemSuccessFactor	0.52	0.52	0.55	0.52	0.52	0.55	0.54	0.53	0.52	0.55	0.53	0.52

Table 19. Comparison of methods' clusters' variable means with production variable dataset.

With the health treatment variable dataset (Table 20) the spread of values is significantly larger when compared to the spread of values in the production variable dataset's clusters. By looking at the *Count*-variable depicting the sizes of clusters, the BIRCH immediately stands out as providing suspicious results by effectively providing something along the lines of outlier separation instead of clustering. Other than that, the other three algorithms seem somewhat aligned in their results:

- Apart from BIRCH, the clustering methods agree with higher production values having a noticeable correlation with numerous treatments for every illness category.
- With *TreatFertilityAvg*, the larger cluster seems to have only a quarter of treatment occasions to the other, the smaller group.
- While ratio isn't similar across the line, the balance coincides with the previous variable for virtually every variable – the smaller cluster seems to experience a larger overall need for treatment than the other.

	BIRCH		HC		SOM		k-Means	
	0	1	0	1	0	1	0	1
Count	817	16	700	133	563	270	653	180
MilkYield	7.451e+05	8.055e+05	7.348e+05	8.064e+05	7.173e+05	8.067e+05	7.275e+05	8.142e+05
MilkYieldMean	9646	9812	9559	1.012e+04	9442	1.008e+04	9536	1.006e+04
ECM	7.385e+05	6.619e+05	7.252e+05	7.995e+05	7.108e+05	7.919e+05	7.171e+05	8.094e+05
ECMMean	9671	9114	9554	1.022e+04	9455	1.009e+04	9528	1.014e+04
FatYield	2.88e+04	2.352e+04	2.814e+04	3.162e+04	2.759e+04	3.1e+04	2.776e+04	3.208e+04
FatMean	380.9	351.2	375	408.8	370.7	400.5	373.5	405.4
ProteinYield	2.421e+04	1.975e+04	2.369e+04	2.639e+04	2.319e+04	2.607e+04	2.338e+04	2.683e+04
ProteinMean	318.5	293.2	313.9	339.8	309.7	335.4	312.6	337.8
FeedingDays	2.751e+04	2.911e+04	2.731e+04	2.877e+04	2.681e+04	2.906e+04	2.709e+04	2.92e+04
SomCellWMean	192.5	159.8	195.3	174.1	195	185.5	194.4	182.7
CowMean	76.23	79.54	75.84	78.66	74.73	79.55	75.29	79.93
UnreliableYield	0.09914	0.125	0.11	0.04511	0.1208	0.05556	0.1149	0.04444
Urea	25.82	22.62	25.61	26.55	25.51	26.27	25.56	26.48
AvgBovineLifetimeMilkYield	2.046e+04	2.427e+04	2.017e+04	2.247e+04	1.981e+04	2.205e+04	2.006e+04	2.225e+04
AvgBovineAge	4.66	5.039	4.651	4.754	4.645	4.713	4.647	4.741
InsemSuccessFactor	0.5301	0.5004	0.5343	0.5042	0.5376	0.5126	0.536	0.5059
TreatFertilityAvg	0.3957	2.184	0.2802	1.219	0.2197	0.8686	0.2462	1.097
TreatMetabolismAvg	0.1151	0.2932	0.1004	0.2134	0.07845	0.202	0.09643	0.1985
TreatDigestionAvg	0.01117	0.0239	0.01003	0.01867	0.007711	0.01913	0.009982	0.0166
TreatUdderAvg	0.2252	0.3033	0.2049	0.3418	0.1506	0.3855	0.1913	0.3553
TreatHoofAvg	0.05972	0.1007	0.0558	0.08525	0.02861	0.127	0.04597	0.1132
TreatRespiratoryAvg	0.005504	0.01542	0.004461	0.01218	0.002098	0.01319	0.003903	0.01219
TreatCalfAvg	0.008923	0.01583	0.007625	0.01658	0.00486	0.01781	0.006765	0.01736

Table 20. Comparison of methods' clusters' variable means with health treatment variable dataset.

After reviewing the cluster means for each method with both training datasets, it is also in place to compare the clustering methods' performances. As already discussed in Chapter 2 under the subchapter Validating the Model, the nature of unsupervised learning, i.e. clustering, is that it lacks a ground truth to which compare the clustering performance to. Therefore, the only way to assess the performance is to compare the clusters by their separation and density and then compare the methods against each other. After calculating the Calinski-Harabaz Index and the Silhouette Coefficient for each method and for both datasets. With the production variable dataset, the scores with both performance evaluation metrics align across the clustering methods (Table 21). If anything, the SOM seems to perform a bit weaker than the other model while *k*-Means seems to offer overall best performance. The Silhouette Coefficients point towards denser and separated clusters in all cases and the Calinski-Harabaz Indices are also close to each other.

	BIRCH	HC	SOM	k-Means
Calinski-Harabaz	1173.64	1173.64	1117.06	1302.71
Silhouette Coeff.	0.55	0.55	0.46	0.54

Table 21. Silhouette Coefficients and Calinski-Harabaz Indices for methods' clusters' density and separation with production variable dataset.

With the health variable dataset, the score spread is notably larger. First off, it is to be noted that the scores for BIRCH disagree with each other – the Silhouette Coefficient is the highest among the clustering methods, while the Calinski-Harabaz Index is the lowest. As already stated, the BIRCH's performance was clearly sub-optimal and the scores seem to confirm that assumption. Other than the BIRCH, the performances of the clustering methods align with the previous dataset, where the SOM's performance was weakest and the *k*-Means' the highest.

	BIRCH	HC	SOM	k-Means
Calinski-Harabaz	238.39	675.09	557.96	750.10
Silhouette Coeff.	0.67	0.53	0.44	0.52

Table 22. Silhouette Coefficients and Calinski-Harabaz Indices for methods' clusters' density and separation with health treatment variable dataset.

Last step was to compare the cluster's populations with each other statistically to see if the clustering methods share similarities between clusters' populations'. This was achieved by running the nonparametric Kruskal-Wallis test, which is a multiple population statistical test with the H_0 hypothesis being that the distributions of the populations are identical [37]. The test was run for each pair of clusters across every method and the variable-wise Kruskal-Wallis tests are given for the production variable dataset and the health treatment variable dataset in Appendices C and D respectively. Because the number of generated comparison tables is large, a scoring matrix was formed to calculate the overall similarity score for a pair of methods. The scoring is calculated as follows:

1. For each cluster in both methods check if there is a single corresponding cluster in the other method by looping through rows and columns using a single variable.
2. For each row and column containing at least a single H-test value equalling or above lesser significance level of 0.01, add a fraction to the final method pair sum.
3. For each row and column containing at least a single H-test value equalling or above higher significance level of 0.05, add another fraction to the final method pair sum.

The size of the fraction depends on the number of clusters, but if a method has matching clusters with another method, using higher significance for a variable the variable-wise

score sum will amount to 1. With lower significance, the sum would total to 0.5. While the method is not exhaustive, it is still informative in the sense that every method is compared to each other in equal terms. However, the case of having two similar clusters for one compared cluster is not addressed; the existence of similarity is only considered.

The Kruskal-Wallis similarity scores are represented in Table 23 for the production variable dataset and Table 24 for the health variable dataset. For the first dataset, there are two distinct groups with distinct similarities. These are the same groups already postulated when looking at the means of clusters' variable distributions – BIRCH with HC and SOM with k -Means. For the second dataset, the highest similarity is between HC and k -Means, which is somewhat visible with the evaluation metrics used earlier. k -Means and SOM share some similarities, as well as BIRCH shares with HC, but nothing as high when compared with the first, the production variable dataset.

	BIRCH	HC	SOM	k-Means
BIRCH	16	16	10	11
HC	16	16	10	11
SOM	10	10	16	15
k-Means	11	11	15	16

Table 23. Kruskal-Wallis H-test similarity scoring matrix with production variable dataset.

	BIRCH	HC	SOM	k-Means
BIRCH	7	5.5	3.5	4.2
HC	5.5	7	4.5	6.2
SOM	3.5	4.5	7	5.2
k-Means	4.2	6.2	5.2	7

Table 24. Kruskal-Wallis H-test similarity scoring matrix with health treatment variable dataset.

To conclude the clustering section, it is advised not to use the BIRCH, as the results were clearly sub-optimal with both datasets. As for the other methods, the choice is ultimately a preference issue. For ease of use, the traditional k -Means and HC methods deliver fair results. However, if the use of a bit more sophisticated SOM is not an obstacle, the output lattice acts well as a division-ratio visualization for the classes and class boundaries. In every case the size of the input dataset with 833 rows seemed sufficient for every method.

4.2 Regression

The second goal for the thesis' machine learning research was to predict select farm performance metrics from others. The metrics are annually measured factors related to farm's operations, finance and performance and constitute the whole dataset, the input and target variables. The research question for this machine learning task is once again a two-fold, albeit more straight-forward, one:

5. *"What model performs best in predicting the select metrics, i.e. target variables?"*
6. *"Are there some input variables with significant explanatory power with regards to the select predicted metrics?"*

As the goal is about predicting real values from the set of data, the suitable machine learning task for attaining the goal is regression. In the terms of machine learning, the input dataset or the independent variables include i.e. milk fat and milk protein percentages, milk profit, food cost, average milk production and farm income. The target dataset or the dependent variables were metrics related to average milk production of both heifers and calvers and number of required artificial insemination services per calving.

When compared with the clustering task, the regression task was much more concise. As opposed to the clustering task, the formation of distinct training and test datasets for differing setups wasn't necessary. The dataset, while sparse, could be easily divided into single datasets for independent and dependent variables. While the data about annual metrics is essentially time-series data, the minuscule frequency directed the research towards using every row as independent input and discarding the time-dimension.

While some of the regression methods allow for multi-output, which means training the model with multiple outputs at the same time, not every method works this way. Therefore, the decision to train each model with a single output was deemed best practice for regression result equivalence and comparability. This, however and as stated, is not mandatory for some of the methods, but merely a way to analogously place the competitors on the same track. Each of the methods were also evaluated with k -Fold cross-validation utilizing R^2 -coefficient of determination and with calculating the mean squared error between the predicted outputs and the ground truths. R^2 -coefficient denotes the degree to which the model can explain the variance of the independents, and a value close to 1 translates to great explanatory power. The mean squared error (MSE) is then the total opposite – the lesser the value, the better. Noteworthy is also the fact that the R^2 -coefficients are the scores of k -Fold cross-validations, while MSEs are calculated from a single training. Therefore the former should be considered as more reliable metric, while the latter is for added insight. For the purposes of visualization, a histogram was produced for each feature showing the distributions of predicted and ground truth values.

4.2.1 Preparing the Dataset

The first part of the dataset preparation was to fetch the correct data from the database backup. This was done with a simple SQL-query to the table labelled *DairyDWFarm* with an organization type and validity filter in place. This however wasn't yet enough, as each row contained only a single annual metric. The conversion to workable dataset was done with a Python script given in Appendix B. After the initial formation of the dataset, there was a total of 92 665 rows for annual farm metrics.

Next step was to take the dataset and divide it to dependent and independent variables. After necessary column drops, the independent dataset contained 163 columns, while the dependent set was constituted of three columns. While only a slice from the ends of the columns, Table 25 depicts well the sparsity of the independent dataset.

	1	2	3	4	6	7	8	9	10	11	...	181	182	183	184	185	186	187	188	189	190
Idkar																					
100001	4.00	3.37	0.0	370.97	1681.00	0.0	21.42	6960.0	13323.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
100001	3.96	3.41	0.0	375.00	1838.45	0.0	23.15	7711.0	14280.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
100001	4.02	3.34	186.0	374.00	2065.51	0.0	20.21	8481.0	14903.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
100001	4.16	3.38	213.0	379.00	2265.50	0.0	18.49	8620.0	13654.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
100001	4.31	3.42	200.0	386.00	2549.99	0.0	16.31	9005.0	16850.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 25. Slices of the dependent dataset for the regression dataset.

A slice of the dependent dataset is depicted subsequently in **Virhe. Viitteen lähde ei löytynyt..** Only one of the features has values, but this is largely due to last two features having values input only after a certain point in time – shown rows are only the first few in the whole dependent variable dataset. While the naming of the independent variables would consume too much space, the independent variables are the following:

- 5: Inseminations per calvings
- 113: Average yield for heifers, kg/cow
- avg_115116: Average yield for calvers, kg/cow (Average of 115 and 116)

	5	113	avg_115116
ldkar			
100001	1.69	0.0	0.0
100001	1.35	0.0	0.0
100001	1.69	0.0	0.0
100001	1.66	0.0	0.0
100001	1.94	0.0	0.0

Table 26 A lookup of first dependent variables, where a row corresponds to an annual value.

Last preparation step was to divide the dependents and independents into training and test sets and normalize and standardize the datasets, both the independent and the dependent one. The scaling of features was achieved by using *StandardScaler*-module found in the *sklearn.preprocessing* Python package. As the row-count of the original dataset was rather excessive, the division to training and test set was applied with respective ratios of 90% and 10% of the dataset. Thus, the training sets contained 83398 rows and the test sets 9267 rows. The scaling, while not required by every regression method, was applied for the comparability of the results.

4.2.2 Ordinary Least Squares

First of the regression machine learning methods was the linear regression method called Ordinary Least Squares (OLS), which has already been discussed in the Chapter 2. The core concept is to define a coefficient for every input variable in a way that minimizes the sum of squared residuals. Residual is the difference between the model's output and the ground truth. Module used was called *LinearRegression* found under *sklearn.linear_model*. The model was fit out-of-the-box, meaning no tuning of the parameters was applied. Like already stated, the OLS was trained separately for each of the three dependent variables and this was the case for each subsequent regression method.

As with all subsequent regression methods, the first of trained models was for feature 5, required inseminations for calvings. With scaled values and for a single run the MSE was 0.2249. The R^2 -coefficient was 0.7691, which is somewhat off the best possible value of 1. The coefficients seem at first glance to provide some useful insights into most explanatory independent features, but the scale for the highest four is off the charts compared to remaining independents. Closer inspection into the raw dataset and these columns show, that the following is true for the outlier-like independent features:

- Feature 35: No values at any of the rows.
- Feature 40: No values at any of the rows.

- Feature 41: No values at any of the rows.
- Feature 72: No values at any of the rows.

As these features were not instructed to be left out by the employer from which the list of features-to-use came, they were left intact in the dataset. As is stated in the conclusions, the regression dataset should be re-constructed with the employer, but more about that later. While their true effect is non-existent, they influence the coefficients for the OLS rendering the information about most explaining features unreliable. (Figure 21)

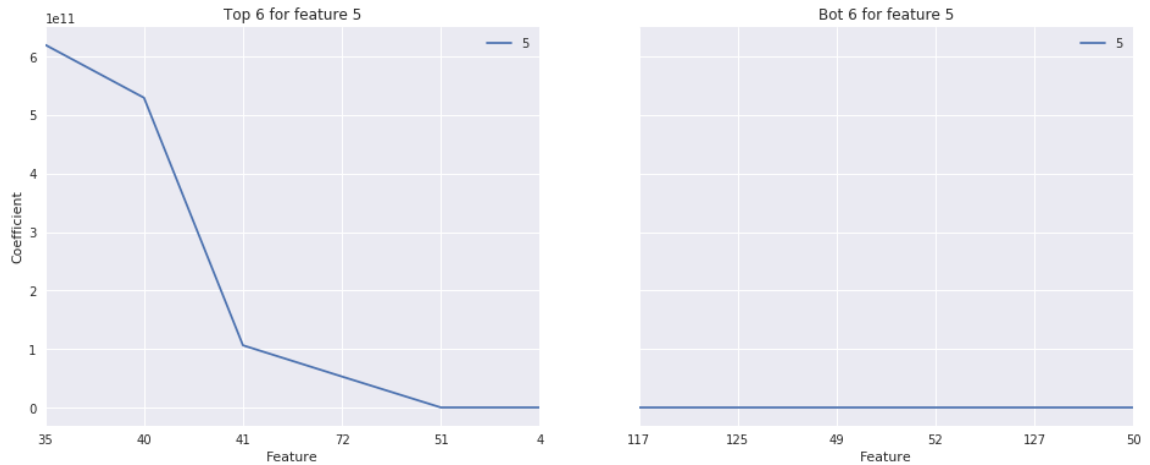


Figure 21. OLS-model's features with highest and lowest coefficients for feature 5.

With the second feature, which is the feature for average bovine-wise heifer milk production, the MSE of 0.0224 and R^2 -coefficient of 0.9783 were significantly better than the for the first feature. The coefficients are however skewed again, but at the other end of the spectrum. The features with extreme coefficients are again the same features with no actual values. Because of the skewness, nothing can be reliably said about the coefficients without inspecting the raw values in the rows first and possibly ditching effectively irrelevant features before training. (Figure 22)

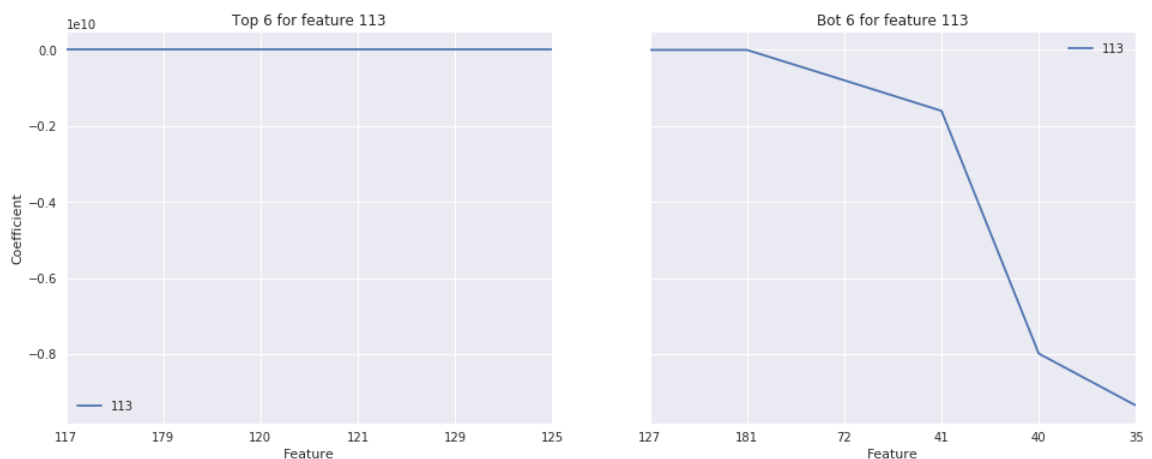


Figure 22. OLS-model's features with highest and lowest coefficients for feature 113.

The last OLS' training was for a feature related to average bovine-wise milk yield for calvers. Astoundingly the MSE resulted to flat zero and the R^2 -coefficient analogously to clean 1. This means, that the model could perfectly predict the outcomes from the input dataset, which is a bit suspicious to say the least. For the last feature, the familiar outlier-coefficients are in line with the other features. The top two features in ranked order are:

- Feature 135: Nitrogen utilisation, %
- Feature 134: Energy ratio MJ/kg ECM, cows in milk

The other features are then very close to zero, having only minuscule to non-existent effect. (Figure 23)

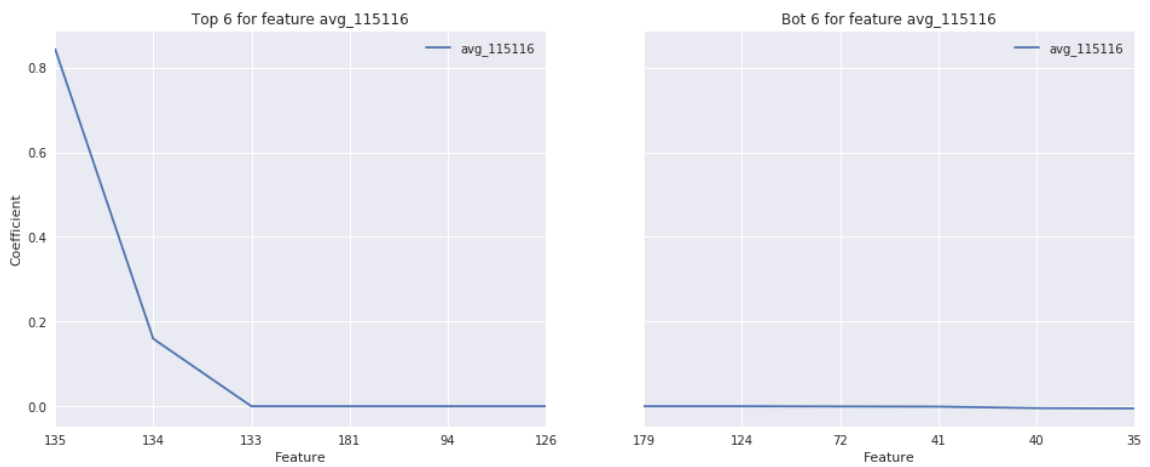


Figure 23. OLS-model's features with highest and lowest coefficients for feature *avg_115116*.

Lastly are the k -Fold cross-validation results for the OLS given in Table 27. To remind, the values are the means and standard deviations of five R^2 -coefficients produced during the cross-validation training and evaluation. While the features for inseminations-per-calving and heifer milk production are rational, the feature for average of calver milk production raises suspicions for a perfect match.

k-Fold CV	OLS		
k = 5	5	113	avg_115116
Mean	0.775006	0.962977	1
Std	0.0059706	0.0300533	0

Table 27. OLS' mean and standard deviation of R^2 -coefficient with five-fold cross-validation.

The overall performance for OLS regarding given two goals was above mediocre but not the best.

4.2.3 Decision Tree Regression

Next regression method taken under scrutiny was a tree-method called Decision Tree Regression (DTR). Discussed in Chapter 2, the tree models attempt splitting the dataset into leaves by finding the equilibrium between the number of leaves and increase of information. The branches (non-endpoint splits) and leaves are essentially rules or feature-wise thresholds, by which the input of features is passed along until a leaf with a satisfied is met. The Python module used in training the method was *DecisionTreeRegressor* found under *sklearn.tree*. The regressor was initialized with default parameters and the most important of those is the criterion for the quality of a split, which is MSE by default.

First training round was again done for feature 5, inseminations per calvings. The MSE resulted in 0.3488 and the R^2 -coefficient in 0.6212. The visualization of features with trees is achieved by calling the trained method's *feature_importances_*-variable and then plotting the values. The biggest importance is given to the following features in order:

- Feature 4: Calving interval
- Feature 51: 12-month rolling services/calving
- Feature 119: Services per calving, heifers
- Feature 50: 12-month rolling calving interval.

These seem much more intuitive than the ones given out by OLS. In fact, they seem to be almost self-evident considering the feature about which the model was trained to give predictions about – inseminations per calving. (Figure 24)

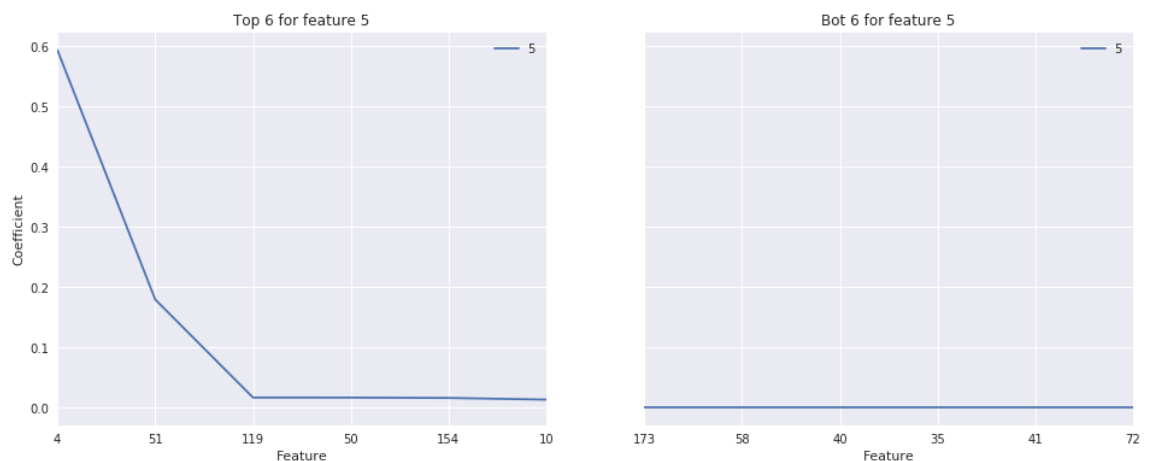


Figure 24. DTR-model's features with highest and lowest coefficients for feature 5.

Second feature, the average yield for heifers, produced results aligned with the OLS. The MSE was a relatively small 0.0266 and the R^2 -coefficient a commendable 0.9729.

The key features for the heifer average milk production were the following:

- Feature 80: Lifetime yield of the culled, kg
- Feature 9: Avg. milk yield kg/cow.

Once again, the key predictive features were closely related to the predicted value, thus producing results void of surprises. The plot for heifer yield feature's input features' importance is given in Figure 25.

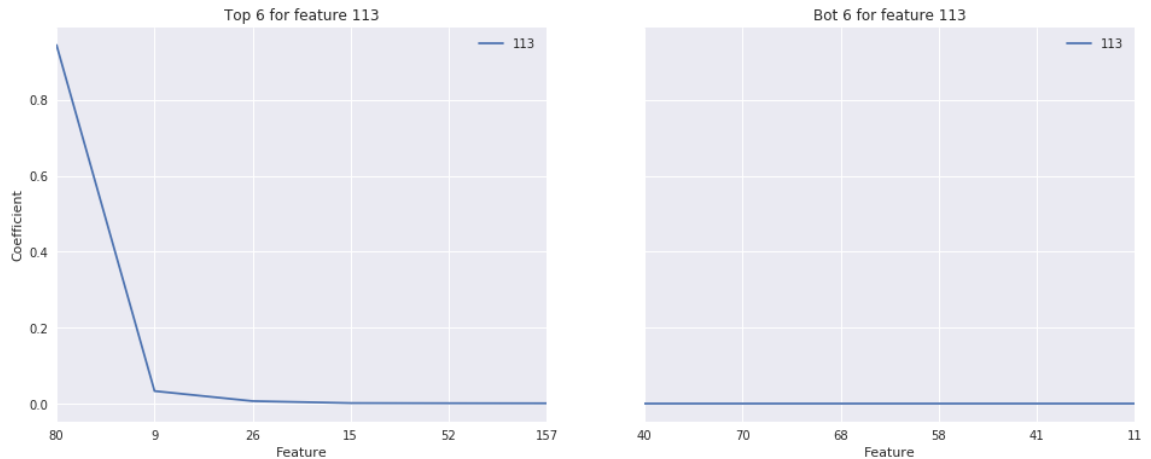


Figure 25. DTR-model's features with highest and lowest coefficients for feature 113.

The trained model for the last of the features, the average yield for calvers, was again in line in the same manner as the OLS. The model seems to be as perfect as the previous one with the MSE being 0 and the R^2 -coefficient 0.9999, which raises suspicions even further. The suspicions are not however related to the model but to the formed dataset, as the result is similar with two distinct regression methods. For the last feature, there is only one significant feature, which is the feature 135, nitrogen utilization-%. All the other features share significances descending from around 1/10000. Figure 26 depicts is information in the form of a plot.

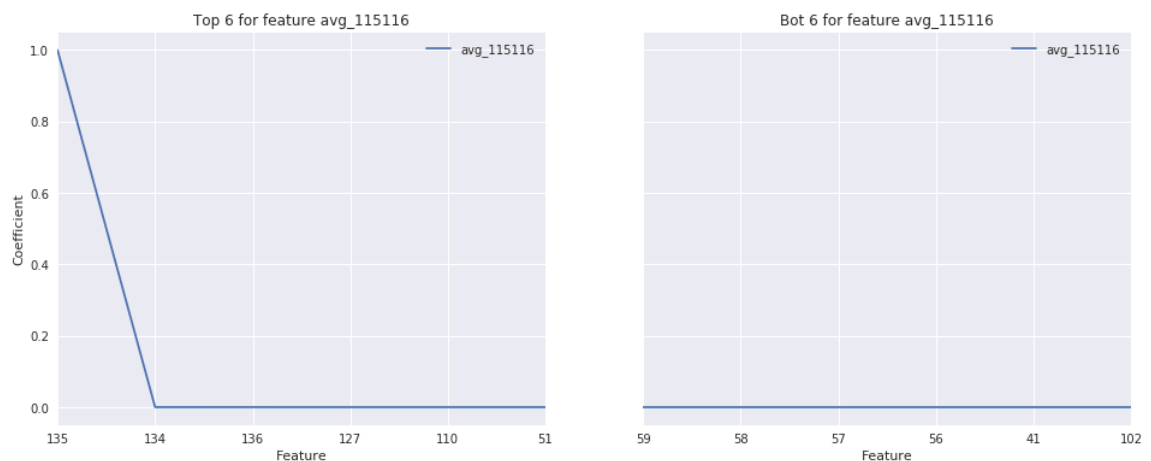


Figure 26. DTR-model's features with highest and lowest coefficients for feature avg_115116.

The five-fold cross-validation yielded results given in Table 28. Once again, the results are the means of R^2 -coefficients. For feature 5, the result is even worse than it was for the OLS. The models for the other two features are however similar, score-wise.

k-Fold CV	DTR		
k = 5	5	113	avg_115116
Mean	0.62	0.972759	0.999709
Std	0.0114602	0.000810766	0.000216599

Table 28. *DTR's mean and standard deviation of R^2 -coefficient with five-fold cross-validation.*

While the DTR performed a bit more poorly than the OLS, the clarity of the most explanatory features far surpasses that of the OLS. With goals being what they are, DTR delivered overall better performance by tackling both given goals at least moderately.

4.2.4 Multilayer Perceptron

The second to last utilized method was Multilayer Perceptron (MLP), which is a deep learning network. The core idea is extensively discussed in Chapter 3. To recapitulate, the MLP is usually formed out of an input layer, one to several hidden layers and an output layer. The layers are linked with weights, that play the key role in MLP learning. The learning happens with a process known as the back-propagation algorithm, which essentially calculates each weight's portion of perceived error between the model's output and the ground truth and then adjusts the weights by a set fraction accordingly. It is also to be mentioned that the DLNs, while able to provide accurate results, lose the descriptiveness of the model quickly due to multi-layered full or partial connectivity. While it has already been stated in previous chapters, deep learning networks require special expert knowledge for model's description. Therefore, the coefficients are omitted for this method.

While the *sklearn*-module also has an MLP, it was deemed better to build the network by utilizing Keras ([38]) as the API and TensorFlow ([39]) as the backend. The reason for this is that this setup allows for seamless utilization of the computer's processor and graphics card, resulting in much faster converging than when trained only with a processor. The network topography constituted of an input layer with 163 input neurons, two hidden layers with 80 hidden neurons in each and an output layer with a single output neuron. For neurons in the hidden layers the activation function was what is known as a rectifier or ramp function for performance reasons discussed in [40]. The single output neuron then had a linear activation function for regression purposes. While options exist, the optimizing function selected was adaptive moment estimation or ADAM, which is an efficient gradient descent optimization algorithm instructed to be used in an in-depth

deep-learning course [41], [42]. The model was trained with batch size of 128 and 32 epochs; increasing the number of epochs would help in achieving greater performance, but at an increased cost of computation with multi-fold cross-validation especially.

While the distribution of predictions was a bit denser than the ground truth, the first training session for the prediction of feature 5 produced significantly better results score-wise than previous regression methods. The MSE for the multilayer network was 0.1767 and the R^2 -coefficient 0.7919.

With the second feature for average production across heifers, the scores stayed relatively exceptional. The MSE was 0.0135 and the R^2 -coefficient was 0.9845. The MLP seems to deliver notably better results than OLS or DTR.

Last of the features, the feature for average milk production across calvers, produced comparable results to other already discussed models. That is, the MSE is almost 0 and the R^2 almost 1.

The cross-validation of the MLP-model produced overall significantly better results than OLS or DTR. Especially the feature for inseminations per calvings experienced impressive relative improvement. The validation results are given in Table 29.

k-Fold CV	MLP		
k = 5	5	113	avg_115116
Mean	0.791852	0.984474	0.999103
Std	0.0351877	0.00115972	0.00144764

Table 29. *MLP's mean and standard deviation of R^2 -coefficient with five-fold cross-validation.*

Overall the MLP produced impressive results score-wise. Goal-wise this is however only half-way there, as the information about features contributing the most to the prediction is lost to the topology of the network.

4.2.5 XGBoost

The last of regression methods was a wild card first introduced to the researcher at an online machine learning course related to machine learning basics with Python ([43]). The extreme gradient boosting trees algorithm, also known as XGBoost, is briefly described under the topic of tree ensembles in Chapter 2. It is a model that has performed exceptionally at multiple online machine learning competitions, such as Kaggle and Netflix related ones ([22]). To recapitulate, the XGBoost is an ensemble of tree models which fits consecutive trees with parameters learned from previous fitted trees. It is an iterative

model, which has been developed specifically to provide accuracy without losing descriptive information and training performance. The Python module used was the *XGBRegressor* found in module *xgboost* found in [44].

Again, the model was first trained to predict the feature for inseminations per calvings. The MSE for the model was 0.1828 and the R^2 -coefficient was 0.8125. Like the DTR, the features are listed in descending importance by the XGBoost regression method. The top five important in order are the following:

- Feature 51: 12-month rolling services/calving
- Feature 4: Calving interval
- Feature 50: 12-month rolling calving interval
- Feature 119: Services per calving, heifers
- Feature 52: Days open

The importance is somewhat more divided between several features rather than just allocated to a single or just a pair of features as was the case with DTR. It also seems that the descent of importance isn't as steep as with other already discussed methods. The plot for feature importances is shown in Figure 27.

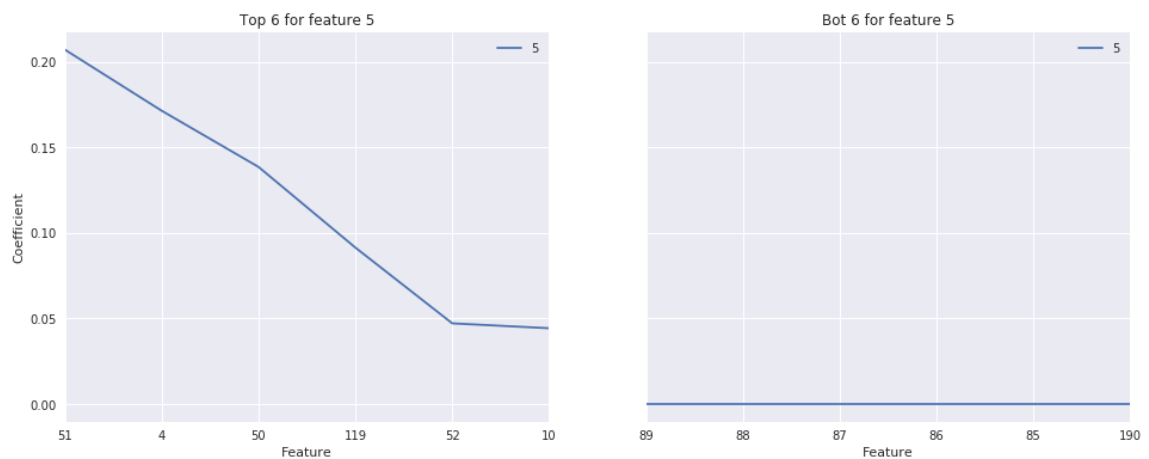


Figure 27. *XGBoost-model's features with highest and lowest coefficients for feature 5.*

Second training run for the independent feature 113 provided also relatively impressive results. The MSE was 0.0144 and the R^2 -coefficient 0.9854. The scores are again similar with the MLP. The importances are again scattered much more across multiple independent features as depicted by the plot in Figure 28. The top six features are:

- Feature 9: Avg. milk yield kg/cow
- Feature 117: ECM yield, kg/cow
- Feature 15: Parity mean
- Feature 26: 12-month rolling milk yield

- Feature 49: Calving age, months
- Feature 157: Heifers culled, %

To underline already made statements, there are multiple self-evident semantical correlations with the independent variable for averages milk production across heifers.

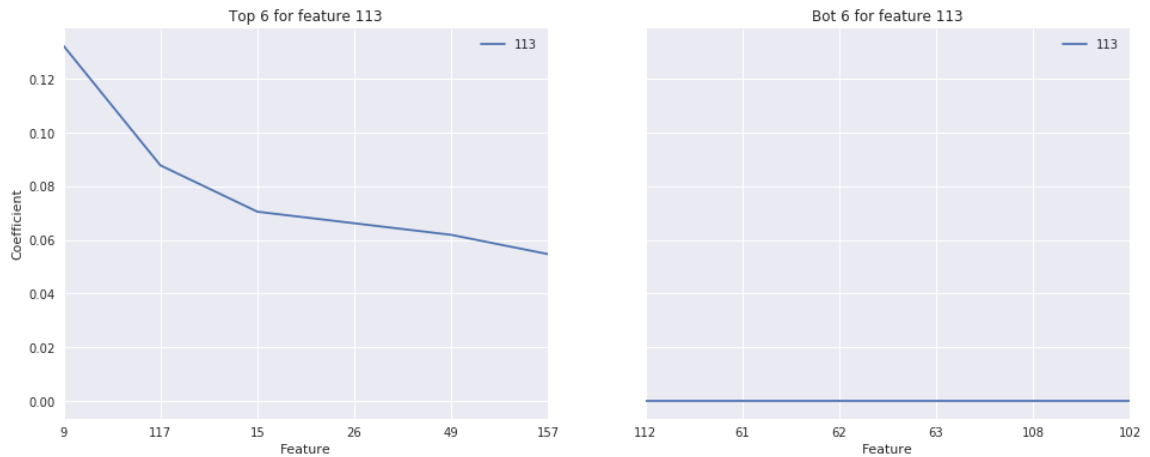


Figure 28. XGBoost-model's features with highest and lowest coefficients for feature 113.

The last of the trained models was for average milk production across all calvers. The scores are within confines of every other model with virtually no distinguishable difference to the MSE and the R^2 -coefficient. Hence the suspicion of dataset or even underlying assumptions about the dataset's formation principles is raised even further. Compared to other two training runs, the importances decline now in more steeper manner. The top two important features for the third independent feature with XGBoost were:

- Feature 135: Nitrogen utilisation, %
- Feature 134: Energy ratio MJ/kg ECM, cows in milk. (Figure 29)

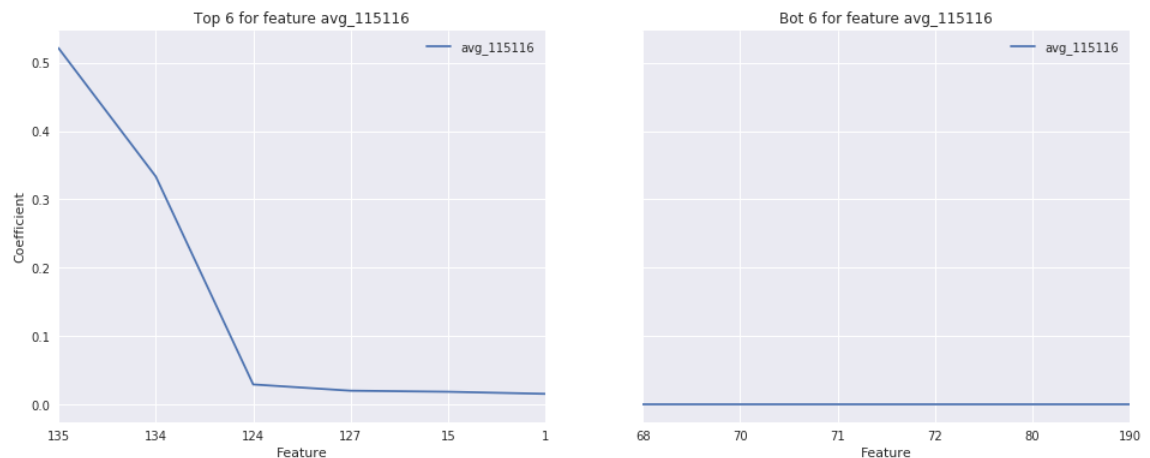


Figure 29. XGBoost-model's features with highest and lowest coefficients for feature avg_115116.

Like for every preceding regression method, the last step was to run the five-fold cross-validation for XGBoost models. The results are depicted in Table 30 and are at least on par with the MLP.

k-Fold CV	XGB		
k = 5	5	113	avg_115116
Mean	0.812532	0.985435	0.999872
Std	0.00571539	0.000366907	0.000186162

Table 30. XGBoost's mean and standard deviation of R^2 -coefficient with five-fold cross-validation.

With similar scores for MSEs and R^2 -coefficients to MLP, the XGBoost actually is the best of all these four methods. The winning factor is the ability to describe the input features in manner of importance to the produced output.

4.2.6 Comparison of Regression Methods

The last step in the comparison of regression models is to compare the trained models by measured performance metrics. The values for both MSE and R^2 -coefficient are given in Table 31 for each method and each feature. Judging only by the scores, the following can be said about handled regression methods:

- DTR performed the poorest out of all methods on average. The MSE was at times double the MLP's value and the method beat the only the OLS with a single feature (113) when scored with R^2 .
- OLS was at times significantly better than DTR, but still fell short on scores when compared to MLP and XGB.
- MLP dominated the MSE-scoring, but couldn't reach the performance of XGB with R^2 .
- XGB was a close second in terms of MSE, but mainly dominated with R^2 -coefficient scores.

As can also be seen from the table, the outcomes for the last of the features (*avg_115116*) shouldn't be considered as reliable. It might be best to either leave the feature out altogether or at least re-think the aggregation for the feature.

		DTR	MLP	OLS	XGB
mse	5	0.345224	0.176743	0.221305	0.182835
	113	0.027243	0.013457	0.020898	0.014354
	avg_115116	0.000010	0.000106	0.000000	0.000024
r2coeff	5	0.620000	0.791852	0.775006	0.812532
	113	0.972759	0.984474	0.962977	0.985435
	avg_115116	0.999709	0.999103	1.000000	0.999872

Table 31. Mean squared errors and R^2 -coefficients for every tested regression method and feature.

The scoring of predictions isn't however the only way to measure the performance and usability, as the goal for this section was a two-part one. The other part was the ability to give insights into most explaining features.

The OLS performed the best time-wise and provided comparatively accurate results. The predictive power of the model was however only one of the measurable attributes regarding set goals. Where the model could deliver satisfactory results with predictions, it performed sub-par in defining meaningful coefficients and was unreliable at best at fading out insignificant features by assigning small coefficients to them. OLS is the optimal regression method, when good predictive power and fast training execution are required over model's reliable explanatory power.

While DTR performed the worst in terms of scores, it provided a massive improvement in terms of reliable explanatory features contrary to the OLS. The descending sets of important input features were intuitive and sensible, thus producing at least moderately satisfactory results for both goals defined for the regression section.

The initial assumption for the MLP was that it would perform relatively well when comparing the predictions to ground truth. This indeed was the case. Unfortunately, the method lacks severely in its ability to provide information about features' importances. The combination of non-linearity paired with full layer-wise connectivity requires extensive reverse-engineering for finding the total coefficients for each method.

The last of the methods taken to comparison was XGBoost, which proved out to be an overall best option. On top of providing scores on par with the MLP, it was also able to distinguish between important and unimportant independent features in human readable manner.

5. CONCLUSIONS

This thesis addressed two questions related to the use of machine learning methods in the context of agricultural data-analysis. These questions were the following:

1. *“What clustering method provides most intuitive model for farm configurations?”*
2. *“What regression model performs best in predicting and explaining select target variables derived from farms’ metrics?”*

After reviewing the relevant theoretical background for traditional machine learning methods and neural networks, the thesis focused on comparing machine learning methods within clustering and regression tasks. The datasets were aggregated from database-backups provided by the employer of the thesis, Mtech Digital Solutions Oy, and the features were chosen according to received instructions as the employer is the expert in the domain of agriculture.

Finnish agrarian entrepreneurs, much akin to entrepreneurs in any sector of business, need both relevant metrics and the ability to interpret them meaningfully. Until now, they have received the analyses from agricultural experts with varying expertise and experience using only the historical data as a basis for their knowledge. The multitude of machine learning methods for data-analysis can tackle this two-fold problem by providing both homogenous and predictive analyses about every farm.

Overall there exists a plethora of machine learning methods to examine and apply for distinct purposes and differing datasets. The division of methods to traditional methods and deep learning networks was made to differentiate between biological inspired learning and algorithmic learning, albeit the division is somewhat artificial. While some of the methods addressed in the theoretical review portion, namely in Chapter 2 and 3, were not taken into comparison, they still were included to help the reader in grasping the broadness of the machine learning and gaining insights into fundamental paradigms at intuitive level. It is also to be noted, that the research in the field of machine learning is constantly advancing with multiple universities and companies taking on massive data mining challenges to best each other.

In addition to machine learning methods as tools for mingling the data and building the models, the understanding of underlying statistical and mathematical principles is somewhat essential for gaining reliable results. The notion of distributions, pitfalls in statistical cross-comparisons, handling of outliers and so forth are necessary steps prior and after the training of a machine learning model. While mathematics was strictly ruled out of scope for this thesis and the focus was discreetly on the intuition-level understanding only, it is advised to familiarize oneself with related mathematical and statistical subjects.

Sources like [1]–[3], [37] accompanied with free online courses on mathematics and machine learning are an especially good starting point.

The theoretical part is however only the first half of the study. The second half, the comparison of both clustering and regression machine learning methods with datasets formed from a database backup provided by MTech Digital Solutions Oy, was indeed the focus of the thesis. For both comparisons, distinct datasets were formed. With both datasets, the independent and dependent variables were defined by the thesis' employer. The dataset for clustering was indeed well-defined, as it was defined and re-defined in mutual conversation with the employer's experts. Same can't be said for the regression dataset however. The regression dataset constituted finally of 163 independent variables, of which many were semantically connected per se. This resulted in regression methods providing often only self-evident information about the most explanatory independent features in respect to predicted dependent feature. There were also columns with null-values, which had an effect of skewing the coefficients with the linear regression method and with many farms the broader book-keeping start usually from mid-2000's.

5.1 Clustering

The first comparison was conducted between the following clustering methods: Hierarchical Clustering, k -Means, Self-Organizing Maps and BIRCH. The research questions for the clustering comparison were the following:

1. *“Are there notable differences related to production between farms and their configurations?”*
2. *“Are there notable differences related to cattle health between farms and their configurations?”*

The dataset for the clustering comparison constituted of cattle-wise production and health treatment data. While the attempt at training all the methods with the same settings was first experimented with, the decision was quickly made to train the methods twice with distinct input datasets. One dataset was for production variables and the other for health-related variables. With production variables, the clustering methods all agreed on differentiating configuration factors when trained with three clusters. With health-related variables the clustering methods agreed on the configurations as well, although this time there was no distinction between the clusters. The distinction was however actually in the production variables, where higher overall production was matched with higher treatment occurrences.

As already stated in the comparison, BIRCH was simply not fit to clustering with the used dataset. The underlying reason might lay in the fact, that it is especially aimed at providing high performance clustering with large datasets. Also, when the number of clusters is

defined by the user, the algorithm uses Hierarchical Clustering instead of tree-like approach. The k -Means and Hierarchical Clustering were intuitive to employ and the results were easy to visualize with the help of dendrograms and plotting of WCSS. The SOM took a bit more effort in terms of setup, but provided intuitive decision boundary visualization accompanied with clusters. In terms of scores, the k -Means had the highest Calinski-Harabaz index with both datasets and relatively equal Silhouette Coefficients. Therefore, the go-to clustering method would be k -Means, according to the comparison and its results.

5.2 Regression

The second comparison was between select regression machine learning methods. The methods chosen for comparison were Ordinary Least Squares (OLS), Decision Tree Regression (DTR), Multilayer Perceptron (MLP) and XGBoost. The research questions for the regression method comparison were:

1. *“What model performs best in predicting the select metrics?”*
2. *“Are there some metrics with significant explanatory power with regards to the select predicted metrics?”*

The dataset for the regression method was formed out of a larger set of farm-related collected metrics, including financial and production-related information. The metrics predicted were about inseminations per calving and average milk production across heifers and calvers separately.

Every regression method performed well in terms of acquiring at least moderate accuracy in terms of mean-squared error and R^2 -coefficient. However, two of the methods, namely MLP and XGBoost, performed notably better than the other two score-wise. The mean-squared errors were lower and the R^2 -coefficients closer to 1 than with OLS and DTR. Thus, MLP and the XGBoost share the podium in respect to the first research question.

The second research question for regression method comparison was however the factor dividing the methods with highest and similar scores. While MLP acquired impressive results, the concept of neural networks is inherently lacking in the ability to describe the importances of input features. While the information is theoretically accessible, the information about features' importances requires expert interpretation accompanied with model's deconstruction, latter of which wasn't accessible in the scope of the thesis. The gradient boosting tree ensemble, e.g. XGBoost, could in turn deliver this information in the same manner as the DTR. Therefore, it outperformed the MLP and emerged as the best option for regression.

5.3 Future research suggestions

First and foremost, the comparison of methods should be re-conducted with re-thought datasets for the regression method comparison for acquisition of relevant information about the dataset. Also, as the machine learning modules were initialized mainly with default settings, an interesting and essential addition to already conducted research would be to tune the parameters of each method for most optimal setup and then conduct the comparison of methods.

The information in the provided database backup would also provide for time-series analysis of the farms and their varying features. As the scope wouldn't allow, this wasn't touched upon within the thesis, but would be an interesting research to conduct as well.

Apart from the subject of machine learning, another research subject would also be to validate the clustering configuration results and then provide an advancement path for farms in terms of farm development for improved balance between higher production and cattle health.

BIBLIOGRAPHY

- [1] P. Flach, *Machine Learning : The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, 2012.
- [2] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*. 2011.
- [3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
- [4] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice-Hall, 1999.
- [5] A. R. Webb and K. D. Copsey, *Statistical Pattern Recognition*. John Wiley & Sons, Ltd, 2011.
- [6] S. V. Kartalopoulos, *Understanding Neural Networks and Fuzzy Logic*. Institute of Electrical and Electronics Engineers, Inc., 1996.
- [7] D. Shiffman, *The Nature of Code*. Self published, 2012.
- [8] T. M. Mitchell, *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.
- [9] N. Intrator, “Competitive Learning,” *Handb. Brain Theory Neural Networks*, pp. 220–223, 1995.
- [10] T. Kohonen, *Self-Organizing Maps*, 3rd ed. Springer Berlin Heidelberg, 2001.
- [11] Y. Frégnac, “Hebbian Synaptic Plasticity: Comparative and Developmental Aspects,” *Handb. Brain Theory Neural Networks*, pp. 459–464, 1995.
- [12] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*. Chapman & Hall/CRC, 1993.
- [13] M. Ljubic and M. Klar, “Estimating expected error rates of random forest classifiers: A comparison of cross-validation and bootstrap,” in *2015 4th Mediterranean Conference on Embedded Computing (MECO)*, 2015, pp. 212–215.
- [14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, and B. Thirion, “Scikit-learn: Machine Learning in Python,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
- [15] P. J. Rousseeuw, “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis.”
- [16] T. Calinski and J. Harabasz, “A dendrite method for cluster analysis,” *Commun. Stat. - Theory Methods*, vol. 3, no. 1, pp. 1–27, 1974.
- [17] C. Moewes and A. Nürnberger, Eds., *Computational Intelligence in Intelligent Data Analysis*, vol. 445. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.
- [18] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 2nd ed. Academic

Press, 2003.

- [19] “Definition of Exemplar by Merriam-Webster.” [Online]. Available: <https://www.merriam-webster.com/dictionary/exemplar>. [Accessed: 31-Jan-2017].
- [20] T. M. Oshiro, P. Santoro Perez, and J. A. Baranauskas, “LNAI 7376 - How Many Trees in a Random Forest?”
- [21] A. Natekin and A. Knoll, “Gradient boosting machines, a tutorial,” *Front. Neurorobot.*, vol. 7, no. DEC, 2013.
- [22] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System.”
- [23] “TOP500 Supercomputer Sites.” [Online]. Available: <https://www.top500.org/>. [Accessed: 24-Jan-2017].
- [24] D. L. Hudson and M. E. Cohen, *Neural Networks and Artificial Intelligence for Biomedical Engineering*. Institute of Electrical and Electronics Engineers, Inc., 2000.
- [25] J. Uusi-Luomalahti, “Classification of Wetland Vegetation Based on TerraSAR-X Data: Comparison of Methods,” Tampere University of Technology, 2016.
- [26] “A Step by Step Backpropagation Example – Matt Mazur.” [Online]. Available: <https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/>. [Accessed: 31-Jan-2017].
- [27] “The 9 Deep Learning Papers You Need To Know About (Understanding CNNs Part 3) – Adit Deshpande – CS Undergrad at UCLA (’19).” [Online]. Available: <https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html>. [Accessed: 01-Feb-2017].
- [28] Y. LeCun and Y. Bengio, “Convolutional Networks for Images, Speech, and Time Series,” *Handb. Brain Theory Neural Networks*, pp. 255–258, 1995.
- [29] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
- [30] D. Lowe, “Radial Basis Function Networks,” *Handb. Brain Theory Neural Networks*, pp. 779–782, 1995.
- [31] “DTREG | RBF Neural Networks.” [Online]. Available: <https://www.dtreg.com/solution/view/25>. [Accessed: 02-Mar-2017].
- [32] G.-J. Jang, M. Kim, Y.-W. Kim, and J. Choi, “Prediction of medical examination results using radial-basis function networks,” in *2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, 2016, pp. 1–4.
- [33] R. P. Shetty, A. Sathyabhama, Srinivasa Pai P, and A. Adarsh Rai, “Optimized Radial Basis Function Neural Network model for wind power prediction,” in *2016 Second International Conference on Cognitive Computing and Information*

Processing (CCIP), 2016, pp. 1–6.

- [34] S.-B. Roh, S.-K. Oh, E. K. Park, and W. Z. Choi, “Design of Radial Basis Function Neural Networks with Principal Component Analysis and Linear Discriminant Analysis for Black Plastic Identification,” in *2016 Joint 8th International Conference on Soft Computing and Intelligent Systems (SCIS) and 17th International Symposium on Advanced Intelligent Systems (ISIS)*, 2016, pp. 764–768.
- [35] H. Ritter, “Self-Organizing Feature Maps: Kohonen Maps,” *Handb. Brain Theory Neural Networks*, pp. 846–851, 1995.
- [36] T. Zhang, R. Ramakrishnan, and M. Livny, “BIRCH: An Efficient Data Clustering Method for Very Large Databases.”
- [37] R. L. Ott and M. Longnecker, *An Introduction to Statistical Methods and Data Analysis*. Brooks/Cole, 2010.
- [38] “Keras Documentation.” [Online]. Available: <https://keras.io/>. [Accessed: 08-May-2017].
- [39] “TensorFlow.” [Online]. Available: <https://www.tensorflow.org/>. [Accessed: 08-May-2017].
- [40] K. He, X. Zhang, S. Ren, and J. Sun, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,” Feb. 2015.
- [41] “SuperDataScience: Deep Learning A-Z™.” [Online]. Available: <https://www.superdatascience.com/deep-learning/>. [Accessed: 08-May-2017].
- [42] D. P. Kingma and J. L. Ba, “ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION.”
- [43] “SuperDataScience: Machine Learning A-Z™.” [Online]. Available: <https://www.superdatascience.com/machine-learning/>. [Accessed: 08-May-2017].
- [44] “XGBoost Documents.” [Online]. Available: <https://xgboost.readthedocs.io/en/latest/>. [Accessed: 09-May-2017].

APPENDIX A: THE SQL-QUERY FOR CLASSIFICATION DATASET

```

SELECT
    -- Independent variables
    IB.Idkar
    ,MRAI.PartOfIdkar
    ,MRAI.MRYear
    ,MRAI.MilkYield
    ,MRAI.MilkYieldMean
    ,MRAI.ECM
    ,MRAI.ECMMean
    ,MRAI.FatYield
    ,MRAI.FatMean
    ,MRAI.ProteinYield
    ,MRAI.ProteinMean
    ,MRAI.FeedingDays
    ,MRAI.SomCellWMean
    ,MRAI.CowMean
    ,MRAI.UnreliableYield
    ,MRAI.Urea
    , (CAST(SUM(MRAB.BovineLifetimeMilkYield) AS FLOAT(24))/CAST(COUNT(IB.BovineId) AS
FLOAT(24))) AS AvgBovineLifetimeMilkYield
    , (CAST(SUM(DATEDIFF(YEAR,B.BirthDate,GETDATE())) AS FLOAT(24))/CAST(COUNT(IB.BovineId) AS
FLOAT(24))) AS AvgBovineAge
    , (CAST(SUM(I.SuccessInseminations) AS FLOAT(24))/CAST(SUM(I.TotalInseminations) AS
FLOAT(24))) AS InsemSuccessFactor
    , (CAST(SUM(HRC0.TreatFertilityCount) AS FLOAT(24))/CAST(COUNT(IB.BovineId) AS FLOAT(24)))
AS TreatFertilityAvg
    , (CAST(SUM(HRC1.TreatMetabolismCount) AS FLOAT(24))/CAST(COUNT(IB.BovineId) AS
FLOAT(24))) AS TreatMetabolismAvg
    , (CAST(SUM(HRC2.TreatDigestionCount) AS FLOAT(24))/CAST(COUNT(IB.BovineId) AS FLOAT(24)))
AS TreatDigestionAvg
    , (CAST(SUM(HRC3.TreatUdderCount) AS FLOAT(24))/CAST(COUNT(IB.BovineId) AS FLOAT(24))) AS
TreatUdderAvg
    , (CAST(SUM(HRC4.TreatHoofCount) AS FLOAT(24))/CAST(COUNT(IB.BovineId) AS FLOAT(24))) AS
TreatHoofAvg
    , (CAST(SUM(HRC5.TreatRespiratoryCount) AS FLOAT(24))/CAST(COUNT(IB.BovineId) AS
FLOAT(24))) AS TreatRespiratoryAvg
    , (CAST(SUM(HRC6.TreatCalfCount) AS FLOAT(24))/CAST(COUNT(IB.BovineId) AS FLOAT(24))) AS
TreatCalfAvg
    -- Configuration variables
    ,IQ1.FarmResult AS IQFeature1
    ,IQ2.FarmResult AS IQFeature2
    ,IQ3.FarmResult AS IQFeature3
    ,IQ4.FarmResult AS IQFeature4
    ,IQ5.FarmResult AS IQFeature5
    FROM ML200_PN.dbo.IdkarBovine as IB
    -- Get Bovine age
    INNER JOIN (SELECT BovineId
    ,BirthDate
    FROM ML200_PN.dbo.Bovine) as B
    ON IB.BovineId = B.BovineId
    -- Get only valid Idkars
    INNER JOIN (SELECT Idkar
    ,OrgParam
    ,ValidDueDate
    FROM ML200_PN.dbo.IdkarCustOrg
    WHERE OrgParam='KATA'
    AND ValidDueDate>GETDATE()) as ICO
    ON ICO.Idkar = IB.Idkar
    -- Get Bovine Lifetime Milk Yield
    INNER JOIN (SELECT BovineId
    ,SUM(MilkYield) AS BovineLifetimeMilkYield
    FROM ML200_PN.dbo.MRAnnualBovine
    GROUP BY BovineId) as MRAB
    ON IB.BovineId = MRAB.BovineId
    -- Get Bovine Insemination Records
    INNER JOIN (SELECT BovineId
    ,COUNT(CASE NRDays WHEN -1 THEN 1 ELSE NULL END) AS SuccessInseminations
    ,COUNT(BovineId) AS TotalInseminations

```

```

,CAST(COUNT(CASE NRDays WHEN -1 THEN 1 ELSE NULL END) as
float(24))/CAST(COUNT(BovineId) AS float(24)) AS InsemSuccessFactor
FROM ML200_PN.dbo.Insemination
GROUP BY BovineId) AS I
ON IB.BovineId = I.BovineId
-- Get Idkar milk production information for whole Idkar (PartOfIdkar=3)
INNER JOIN (SELECT Idkar
,MYear
,PartOfIdkar
,MilkYield
,MilkYieldMean
,ECM
,ECMMean
,FatYield
,FatMean
,ProteinYield
,ProteinMean
,FeedingDays
,SomCellWMean
,CowMean
,UnreliableYield
,Urea
FROM ML200_PN.dbo.MRAnnualIdkar
WHERE (
--MYear = 2015
--OR
MYear = 2016
)
AND PartOfIdkar = 3) as MRAI
ON ICO.Idkar = MRAI.Idkar
-- Get HealthCodes sorted into categories
-- Joins are FULL, as not every Bovine have a treatment record on every category
FULL JOIN(SELECT BovineId
,COUNT(HealthCode) AS TreatFertilityCount
FROM ML200_PN.dbo.HealthRecCattle
WHERE HealthCode > 9
AND HealthCode < 63
GROUP BY BovineId) AS HRC0
ON IB.BovineId = HRC0.BovineId
FULL JOIN(SELECT BovineId
,COUNT(HealthCode) AS TreatMetabolismCount
FROM ML200_PN.dbo.HealthRecCattle
WHERE HealthCode > 99
AND HealthCode < 141
GROUP BY BovineId) AS HRC1
ON IB.BovineId = HRC1.BovineId
FULL JOIN(SELECT BovineId
,COUNT(HealthCode) AS TreatDigestionCount
FROM ML200_PN.dbo.HealthRecCattle
WHERE HealthCode > 209
AND HealthCode < 244
GROUP BY BovineId) AS HRC2
ON IB.BovineId = HRC2.BovineId
FULL JOIN(SELECT BovineId
,COUNT(HealthCode) AS TreatUdderCount
FROM ML200_PN.dbo.HealthRecCattle
WHERE HealthCode > 299
AND HealthCode < 304
GROUP BY BovineId) AS HRC3
ON IB.BovineId = HRC3.BovineId
FULL JOIN(SELECT BovineId
,COUNT(HealthCode) AS TreatHoofCount
FROM ML200_PN.dbo.HealthRecCattle
WHERE HealthCode > 350
AND HealthCode < 370
GROUP BY BovineId) AS HRC4
ON IB.BovineId = HRC4.BovineId
FULL JOIN(SELECT BovineId
,COUNT(HealthCode) AS TreatRespiratoryCount
FROM ML200_PN.dbo.HealthRecCattle
WHERE HealthCode = 400
OR HealthCode = 401
OR HealthCode = 403
GROUP BY BovineId) AS HRC5

```

```

ON IB.BovineId = HRC5.BovineId
FULL JOIN(SELECT BovineId
          ,COUNT(HealthCode) AS TreatCalfCount
          FROM ML200_PN.dbo.HealthRecCattle
          WHERE HealthCode = 252
          OR HealthCode = 402
          GROUP BY BovineId) AS HRC6
ON IB.BovineId = HRC6.BovineId
-- Get valid dependent IdkarQuality fields as separate columns
INNER JOIN (SELECT Idkar
            ,IQFeature
            ,FarmResult
            FROM ML200_PN.dbo.IdkarQuality
            WHERE IQFeature = 22
            AND ML200_PN.dbo.IdkarQuality.ValidDueDate>GETDATE()) as IQ1
ON ICO.Idkar = IQ1.Idkar
INNER JOIN (SELECT Idkar
            ,IQFeature
            ,FarmResult
            FROM ML200_PN.dbo.IdkarQuality
            WHERE IQFeature = 32
            AND ML200_PN.dbo.IdkarQuality.ValidDueDate>GETDATE()) as IQ2
ON ICO.Idkar = IQ2.Idkar
INNER JOIN (SELECT Idkar
            ,IQFeature
            ,FarmResult
            FROM ML200_PN.dbo.IdkarQuality
            WHERE IQFeature = 36
            AND ML200_PN.dbo.IdkarQuality.ValidDueDate>GETDATE()) as IQ3
ON ICO.Idkar = IQ3.Idkar
INNER JOIN (SELECT Idkar
            ,IQFeature
            ,FarmResult
            FROM ML200_PN.dbo.IdkarQuality
            WHERE IQFeature = 47
            AND ML200_PN.dbo.IdkarQuality.ValidDueDate>GETDATE()) as IQ4
ON ICO.Idkar = IQ4.Idkar
INNER JOIN (SELECT Idkar
            ,IQFeature
            ,FarmResult
            FROM ML200_PN.dbo.IdkarQuality
            WHERE IQFeature = 57
            AND ML200_PN.dbo.IdkarQuality.ValidDueDate>GETDATE()) as IQ5
ON ICO.Idkar = IQ5.Idkar
-- Get only valid Bovines from IdkarBovine
WHERE IB.ValidDueDate>GETDATE()
AND IB.BovineRole=1

GROUP BY IB.Idkar
        ,MRAI.PartOfIdkar
        ,MRAI.MRYear
        ,MRAI.MilkYield
        ,MRAI.MilkYieldMean
        ,MRAI.ECM
        ,MRAI.ECMMean
        ,MRAI.FatYield
        ,MRAI.FatMean
        ,MRAI.ProteinYield
        ,MRAI.ProteinMean
        ,MRAI.FeedingDays
        ,MRAI.SomCellWMean
        ,MRAI.CowMean
        ,MRAI.UnreliableYield
        ,MRAI.Urea
        ,IQ1.FarmResult
        ,IQ2.FarmResult
        ,IQ3.FarmResult
        ,IQ4.FarmResult
        ,IQ5.FarmResult
ORDER BY IB.Idkar

```

APPENDIX B: PYTHON NOTEBOOK CODE FOR REGRESSION DATASET

```
In [ ]: import pandas as pd
        from IPython.display import display
```

Import valid DairyDWFarm rows,

The data is originally a direct CSV-import from an SQL-query with only valid Idkars selected.

```
In [ ]: data = pd.read_csv('DairyDWFarm_Filtered.csv', delimiter=';', )
        display(data.head())
```

Pivot the Feature values yearly for each Idkar

Last step is to form a matrix of FarmResults. Each row contains yearly information for a single Idkar. The columns are only for Feature numbers, others were left out. ResultValidity was 1 when the value for field was NULL, so the value of the field is an implication of validity in itself. Thus the column is also omitted. Other dropped fields were BmProdType, FarmTarget, Modifier and ModificationDate.

```
In [ ]: data = data.pivot_table(index=['Idkar', 'StatYear'],
                                columns='Feature',
                                values='FarmResult')
        data.fillna(value=0, inplace=True)
        display(data.head())
```

Form two columns for average yields, one for avg. of heifers and one for avg. of all calvers.

```
In [ ]: calvers_avg = pd.Series(index=data.index)
        for row in data.itertuples():
            avg = (row._115+row._116)/2
            calvers_avg.loc[row.Index] = avg
        data = data.assign(avg_115116=calvers_avg)
```

Save the Dataset to a file

```
In [ ]: data.to_csv('Dataset_Regression.csv', sep=';')
```

APPENDIX C: KRUSKAL-WALLIS H-TESTS FOR PRODUCTION VARIABLE DATASET'S CLUSTERS

Kruskal-Wallis for MilkYield

	BIRCH			HC			SOM			k-Means			
	0	1	2	0	1	2	0	1	2	0	1	2	
BIRCH	0	1	2.7e-80	1.13e-58	1	2.7e-80	1.13e-58	4.44e-76	0.0136	1.81e-66	4.89e-79	0.00183	1.79e-63
	1	2.7e-80	1	5.46e-84	2.7e-80	1	5.46e-84	0.000115	2.84e-70	2.21e-82	0.000871	3.93e-70	1.38e-91
	2	1.13e-58	5.46e-84	1	1.13e-58	5.46e-84	1	1.16e-82	8.04e-53	5.95e-06	6.32e-85	1.3e-52	0.013
HC	0	1	2.7e-80	1.13e-58	1	2.7e-80	1.13e-58	4.44e-76	0.0136	1.81e-66	4.89e-79	0.00183	1.79e-63
	1	2.7e-80	1	5.46e-84	2.7e-80	1	5.46e-84	0.000115	2.84e-70	2.21e-82	0.000871	3.93e-70	1.38e-91
	2	1.13e-58	5.46e-84	1	1.13e-58	5.46e-84	1	1.16e-82	8.04e-53	5.95e-06	6.32e-85	1.3e-52	0.013
SOM	0	4.44e-76	0.000115	1.16e-82	4.44e-76	0.000115	1.16e-82	1	4.08e-68	9.7e-86	0.504	4.05e-68	1.58e-90
	1	0.0136	2.84e-70	8.04e-53	0.0136	2.84e-70	8.04e-53	4.08e-68	1	2.26e-59	6.56e-71	0.485	1.15e-56
	2	1.81e-66	2.21e-82	5.95e-06	1.81e-66	2.21e-82	5.95e-06	9.7e-86	2.26e-59	1	6.33e-89	3.29e-59	0.00886

Kruskal-Wallis for MilkYieldMean

	BIRCH			HC			SOM			k-Means			
	0	1	2	0	1	2	0	1	2	0	1	2	
BIRCH	0	1	0.000559	1.36e-27	1	0.000559	1.36e-27	0.157	0.855	1.08e-35	0.00659	0.773	4.15e-29
	1	0.000559	1	6.92e-29	0.000559	1	6.92e-29	0.00376	0.000638	3.38e-40	0.281	0.000467	7.04e-31
	2	1.36e-27	6.92e-29	1	1.36e-27	6.92e-29	1	1.47e-36	1.35e-25	0.904	7.6e-32	1.38e-25	0.595
HC	0	1	0.000559	1.36e-27	1	0.000559	1.36e-27	0.157	0.855	1.08e-35	0.00659	0.773	4.15e-29
	1	0.000559	1	6.92e-29	0.000559	1	6.92e-29	0.00376	0.000638	3.38e-40	0.281	0.000467	7.04e-31
	2	1.36e-27	6.92e-29	1	1.36e-27	6.92e-29	1	1.47e-36	1.35e-25	0.904	7.6e-32	1.38e-25	0.595
SOM	0	0.157	0.00376	1.47e-36	0.157	0.00376	1.47e-36	1	0.125	5.26e-51	0.065	0.101	1.01e-39
	1	0.855	0.000638	1.35e-25	0.855	0.000638	1.35e-25	0.125	1	1.38e-32	0.00613	0.912	8.43e-27
	2	1.08e-35	3.38e-40	0.904	1.08e-35	3.38e-40	0.904	5.26e-51	1.38e-32	1	1.7e-44	1.8e-32	0.629

Kruskal-Wallis for ECM

	BIRCH			HC			SOM			k-Means			
	0	1	2	0	1	2	0	1	2	0	1	2	
BIRCH	0	1	6.02e-84	1.13e-58	1	6.02e-84	1.13e-58	4.27e-80	0.00775	4.57e-67	1.46e-81	0.00585	1.76e-63
	1	6.02e-84	1	1.13e-81	6.02e-84	1	1.13e-81	1.48e-05	2.58e-72	2.68e-85	0.000362	8.41e-70	1.67e-89
	2	1.13e-58	1.13e-81	1	1.13e-58	1.13e-81	1	2.06e-81	8.04e-53	1.79e-05	1.87e-82	4.11e-52	0.0102
HC	0	1	6.02e-84	1.13e-58	1	6.02e-84	1.13e-58	4.27e-80	0.00775	4.57e-67	1.46e-81	0.00585	1.76e-63
	1	6.02e-84	1	1.13e-81	6.02e-84	1	1.13e-81	1.48e-05	2.58e-72	2.68e-85	0.000362	8.41e-70	1.67e-89
	2	1.13e-58	1.13e-81	1	1.13e-58	1.13e-81	1	2.06e-81	8.04e-53	1.79e-05	1.87e-82	4.11e-52	0.0102
SOM	0	4.27e-80	1.48e-05	2.06e-81	4.27e-80	1.48e-05	2.06e-81	1	7.51e-71	3.42e-90	0.365	1.89e-68	1.17e-89
	1	0.00775	2.58e-72	8.04e-53	0.00775	2.58e-72	8.04e-53	7.51e-71	1	1.73e-59	5.12e-72	0.895	1.15e-56
	2	4.57e-67	2.68e-85	1.79e-05	4.57e-67	2.68e-85	1.79e-05	3.42e-90	1.73e-59	1	3.56e-92	2.13e-58	0.0215

Kruskal-Wallis for ECMMean

	BIRCH			HC			SOM			k-Means			
	0	1	2	0	1	2	0	1	2	0	1	2	
BIRCH	0	1	0.000572	1.14e-22	1	0.000572	1.14e-22	0.15	0.835	5.25e-31	0.00748	0.958	3.4e-24
	1	0.000572	1	4.05e-20	0.000572	1	4.05e-20	0.00617	0.000678	3.86e-30	0.249	0.00207	6.95e-22
	2	1.14e-22	4.05e-20	1	1.14e-22	4.05e-20	1	2.83e-26	2.91e-21	0.87	4.35e-23	2.3e-19	0.67
HC	0	1	0.000572	1.14e-22	1	0.000572	1.14e-22	0.15	0.835	5.25e-31	0.00748	0.958	3.4e-24
	1	0.000572	1	4.05e-20	0.000572	1	4.05e-20	0.00617	0.000678	3.86e-30	0.249	0.00207	6.95e-22
	2	1.14e-22	4.05e-20	1	1.14e-22	4.05e-20	1	2.83e-26	2.91e-21	0.87	4.35e-23	2.3e-19	0.67
SOM	0	0.15	0.00617	2.83e-26	0.15	0.00617	2.83e-26	1	0.126	4.02e-39	0.103	0.197	4.33e-29
	1	0.835	0.000678	2.91e-21	0.835	0.000678	2.91e-21	0.126	1	1.18e-28	0.00722	0.891	1.49e-22
	2	5.25e-31	3.86e-30	0.87	5.25e-31	3.86e-30	0.87	4.02e-39	1.18e-28	1	1.4e-34	5.67e-26	0.495

Kruskal-Wallis for FatYield

	BIRCH			HC			SOM			k-Means			
	0	1	2	0	1	2	0	1	2	0	1	2	
BIRCH	0	1	1.72e-78	6.48e-57	1	1.72e-78	6.48e-57	6.61e-75	0.00949	1.6e-64	6.62e-76	0.0109	3.06e-61
	1	1.72e-78	1	3.04e-70	1.72e-78	1	3.04e-70	6.58e-05	1.94e-70	2.9e-74	0.000494	1.6e-62	3.65e-76
	2	6.48e-57	3.04e-70	1	6.48e-57	3.04e-70	1	1.12e-70	1.42e-52	0.000129	1.27e-72	5.15e-48	0.0168
HC	0	1	1.72e-78	6.48e-57	1	1.72e-78	6.48e-57	6.61e-75	0.00949	1.6e-64	6.62e-76	0.0109	3.06e-61
	1	1.72e-78	1	3.04e-70	1.72e-78	1	3.04e-70	6.58e-05	1.94e-70	2.9e-74	0.000494	1.6e-62	3.65e-76
	2	6.48e-57	3.04e-70	1	6.48e-57	3.04e-70	1	1.12e-70	1.42e-52	0.000129	1.27e-72	5.15e-48	0.0168
SOM	0	6.61e-75	6.58e-05	1.12e-70	6.61e-75	6.58e-05	1.12e-70	1	7.18e-69	5.72e-79	0.554	2.8e-61	1.16e-77
	1	0.00949	1.94e-70	1.42e-52	0.00949	1.94e-70	1.42e-52	7.18e-69	1	6.36e-59	7.85e-70	0.955	3.2e-56
	2	1.6e-64	2.9e-74	0.000129	1.6e-64	2.9e-74	0.000129	5.72e-79	6.36e-59	1	2.45e-81	1.85e-53	0.07

Kruskal-Wallis for FatMean

	BIRCH			HC			SOM			k-Means			
	0	1	2	0	1	2	0	1	2	0	1	2	
BIRCH	0	1	0.018	1.22e-13	1	0.018	1.22e-13	0.679	0.977	6.74e-21	0.119	0.859	1.76e-15
	1	0.018	1	6.44e-12	0.018	1	6.44e-12	0.00846	0.0251	6.03e-21	0.25	0.0571	3.5e-14
	2	1.22e-13	6.44e-12	1	1.22e-13	6.44e-12	1	7.3e-17	1.08e-12	0.442	2.66e-14	4.03e-11	0.948
HC	0	1	0.018	1.22e-13	1	0.018	1.22e-13	0.679	0.977	6.74e-21	0.119	0.859	1.76e-15
	1	0.018	1	6.44e-12	0.018	1	6.44e-12	0.00846	0.0251	6.03e-21	0.25	0.0571	3.5e-14
	2	1.22e-13	6.44e-12	1	1.22e-13	6.44e-12	1	7.3e-17	1.08e-12	0.442	2.66e-14	4.03e-11	0.948
SOM	0	0.679	0.00846	7.3e-17	0.679	0.00846	7.3e-17	1	0.687	1.43e-28	0.127	0.87	3.58e-20
	1	0.977	0.0251	1.08e-12	0.977	0.0251	1.08e-12	0.687	1	3.54e-19	0.138	0.843	2.6e-14
	2	6.74e-21	6.03e-21	0.442	6.74e-21	6.03e-21	0.442	1.43e-28	3.54e-19	1	9.12e-25	9.08e-17	0.36

Kruskal-Wallis for ProteinYield

	BIRCH			HC			SOM			k-Means			
	0	1	2	0	1	2	0	1	2	0	1	2	
BIRCH	0	1	1.5e-78	5.85e-57	1	1.5e-78	5.85e-57	3.91e-75	0.00811	1.47e-64	4.09e-76	0.00946	2.83e-61
	1	1.5e-78	1	9.86e-71	1.5e-78	1	9.86e-71	4.9e-05	1.96e-70	1.74e-75	0.000455	1.61e-62	7.02e-77
	2	5.85e-57	9.86e-71	1	5.85e-57	9.86e-71	1	4.81e-71	1.29e-52	0.000112	7.33e-73	4.88e-48	0.014
HC	0	1	1.5e-78	5.85e-57	1	1.5e-78	5.85e-57	3.91e-75	0.00811	1.47e-64	4.09e-76	0.00946	2.83e-61
	1	1.5e-78	1	9.86e-71	1.5e-78	1	9.86e-71	4.9e-05	1.96e-70	1.74e-75	0.000455	1.61e-62	7.02e-77
	2	5.85e-57	9.86e-71	1	5.85e-57	9.86e-71	1	4.81e-71	1.29e-52	0.000112	7.33e-73	4.88e-48	0.014
SOM	0	3.91e-75	4.9e-05	4.81e-71	3.91e-75	4.9e-05	4.81e-71	1	4.71e-69	3.55e-80	0.523	2.21e-61	3.14e-78
	1	0.00811	1.96e-70	1.29e-52	0.00811	1.96e-70	1.29e-52	4.71e-69	1	5.97e-59	4.65e-70	0.939	2.99e-56
	2	1.47e-64	1.74e-75	0.000112	1.47e-64	1.74e-75	0.000112	3.55e-80	5.97e-59	1	1.91e-82	1.77e-53	0.0719

Kruskal-Wallis for ProteinMean

	BIRCH			HC			SOM			k-Means			
	0	1	2	0	1	2	0	1	2	0	1	2	
BIRCH	0	1	0.000858	3.1e-23	1	0.000858	3.1e-23	0.16	0.751	3.45e-31	0.0105	0.936	8.02e-25
	1	0.000858	1	1.13e-20	0.000858	1	1.13e-20	0.00771	0.000586	1.05e-30	0.25	0.00242	1.18e-22
	2	3.1e-23	1.13e-20	1	3.1e-23	1.13e-20	1	1.75e-27	5.02e-22	0.711	5.48e-24	1.85e-19	0.7
HC	0	1	0.000858	3.1e-23	1	0.000858	3.1e-23	0.16	0.751	3.45e-31	0.0105	0.936	8.02e-25
	1	0.000858	1	1.13e-20	0.000858	1	1.13e-20	0.00771	0.000586	1.05e-30	0.25	0.00242	1.18e-22
	2	3.1e-23	1.13e-20	1	3.1e-23	1.13e-20	1	1.75e-27	5.02e-22	0.711	5.48e-24	1.85e-19	0.7
SOM	0	0.16	0.00771	1.75e-27	0.16	0.00771	1.75e-27	1	0.0984	2.72e-40	0.12	0.183	1.53e-30
	1	0.751	0.000586	5.02e-22	0.751	0.000586	5.02e-22	0.0984	1	4.47e-29	0.00637	0.827	2.27e-23
	2	3.45e-31	1.05e-30	0.711	3.45e-31	1.05e-30	0.711	2.72e-40	4.47e-29	1	1.67e-35	1.11e-25	0.405

Kruskal-Wallis for FeedingDays

	BIRCH			HC			SOM			k-Means			
	0	1	2	0	1	2	0	1	2	0	1	2	
BIRCH	0	1	3.85e-78	1.26e-58	1	3.85e-78	1.26e-58	1.74e-74	0.0206	1.08e-64	1.77e-76	0.00349	3.85e-63
	1	3.85e-78	1	1.84e-74	3.85e-78	1	1.84e-74	0.01	6.14e-69	2.49e-60	0.00331	8.66e-69	1.07e-77
	2	1.26e-58	1.84e-74	1	1.26e-58	1.84e-74	1	2.12e-73	8.04e-53	1.29e-05	8.65e-77	1.3e-52	0.0342
HC	0	1	3.85e-78	1.26e-58	1	3.85e-78	1.26e-58	1.74e-74	0.0206	1.08e-64	1.77e-76	0.00349	3.85e-63
	1	3.85e-78	1	1.84e-74	3.85e-78	1	1.84e-74	0.01	6.14e-69	2.49e-60	0.00331	8.66e-69	1.07e-77
	2	1.26e-58	1.84e-74	1	1.26e-58	1.84e-74	1	2.12e-73	8.04e-53	1.29e-05	8.65e-77	1.3e-52	0.0342
SOM	0	1.74e-74	0.01	2.12e-73	1.74e-74	0.01	2.12e-73	1	4.45e-67	2.61e-62	0.802	4.93e-67	5.1e-77
	1	0.0206	6.14e-69	8.04e-53	0.0206	6.14e-69	8.04e-53	4.45e-67	1	5.38e-58	4.89e-69	0.51	1.57e-56
	2	1.08e-64	2.49e-60	1.29e-05	1.08e-64	2.49e-60	1.29e-05	2.61e-62	5.38e-58	1	2.28e-66	7.55e-58	0.00779

Kruskal-Wallis for SomCellWMean

	BIRCH			HC			SOM			k-Means			
	0	1	2	0	1	2	0	1	2	0	1	2	
BIRCH	0	1	0.05	4.33e-05	1	0.05	4.33e-05	0.143	0.916	1.5e-05	0.0805	0.831	0.000116
	1	0.05	1	0.001	0.05	1	0.001	0.547	0.0547	0.000632	0.748	0.134	0.00315
	2	4.33e-05	0.001	1	4.33e-05	0.001	1	0.000279	6.27e-05	0.8	0.000418	0.00038	0.665
HC	0	1	0.05	4.33e-05	1	0.05	4.33e-05	0.143	0.916	1.5e-05	0.0805	0.831	0.000116
	1	0.05	1	0.001	0.05	1	0.001	0.547	0.0547	0.000632	0.748	0.134	0.00315
	2	4.33e-05	0.001	1	4.33e-05	0.001	1	0.000279	6.27e-05	0.8	0.000418	0.00038	0.665
SOM	0	0.143	0.547	0.000279	0.143	0.547	0.000279	1	0.144	0.000134	0.765	0.29	0.000879
	1	0.916	0.0547	6.27e-05	0.916	0.0547	6.27e-05	0.144	1	2.67e-05	0.0846	0.761	0.000166
	2	1.5e-05	0.000632	0.8	1.5e-05	0.000632	0.8	0.000134	2.67e-05	1	0.000218	0.000196	0.835

Kruskal-Wallis for CowMean

	BIRCH			HC			SOM			k-Means			
	0	1	2	0	1	2	0	1	2	0	1	2	
BIRCH	0	1	5.98e-78	1.46e-57	1	5.98e-78	1.46e-57	1.77e-74	0.0207	1.68e-63	2.81e-76	0.00353	3.6e-62
	1	5.98e-78	1	1.47e-64	5.98e-78	1	1.47e-64	0.0167	8.88e-69	3.46e-51	0.00437	1.27e-68	1.86e-67
	2	1.46e-57	1.47e-64	1	1.46e-57	1.47e-64	1	5.15e-64	2.86e-52	4.76e-05	3.03e-67	4.14e-52	0.0573
HC	0	1	5.98e-78	1.46e-57	1	5.98e-78	1.46e-57	1.77e-74	0.0207	1.68e-63	2.81e-76	0.00353	3.6e-62
	1	5.98e-78	1	1.47e-64	5.98e-78	1	1.47e-64	0.0167	8.88e-69	3.46e-51	0.00437	1.27e-68	1.86e-67
	2	1.46e-57	1.47e-64	1	1.46e-57	1.47e-64	1	5.15e-64	2.86e-52	4.76e-05	3.03e-67	4.14e-52	0.0573
SOM	0	1.77e-74	0.0167	5.15e-64	1.77e-74	0.0167	5.15e-64	1	4.34e-67	3.18e-53	0.726	4.8e-67	2.79e-67
	1	0.0207	8.88e-69	2.86e-52	0.0207	8.88e-69	2.86e-52	4.34e-67	1	3.02e-57	7.3e-69	0.511	4.93e-56
	2	1.68e-63	3.46e-51	4.76e-05	1.68e-63	3.46e-51	4.76e-05	3.18e-53	3.02e-57	1	4.93e-57	3.94e-57	0.0108

Kruskal-Wallis for UnreliableYield

	BIRCH			HC			SOM			k-Means			
	0	1	2	0	1	2	0	1	2	0	1	2	
BIRCH	0	1	0.0184	2.12e-07	1	0.0184	2.12e-07	0.074	0.378	1.4e-07	0.0437	0.509	6.86e-07
	1	0.0184	1	4.3e-06	0.0184	1	4.3e-06	0.349	0.00471	1.02e-06	0.56	0.121	1.56e-05
	2	2.12e-07	4.3e-06	1	2.12e-07	4.3e-06	1	1.07e-07	1.36e-07	0.952	3.63e-07	1.68e-05	0.712
HC	0	1	0.0184	2.12e-07	1	0.0184	2.12e-07	0.074	0.378	1.4e-07	0.0437	0.509	6.86e-07
	1	0.0184	1	4.3e-06	0.0184	1	4.3e-06	0.349	0.00471	1.02e-06	0.56	0.121	1.56e-05
	2	2.12e-07	4.3e-06	1	2.12e-07	4.3e-06	1	1.07e-07	1.36e-07	0.952	3.63e-07	1.68e-05	0.712
SOM	0	0.074	0.349	1.07e-07	0.074	0.349	1.07e-07	1	0.0178	2.06e-08	0.715	0.323	4.4e-07
	1	0.378	0.00471	1.36e-07	0.378	0.00471	1.36e-07	0.0178	1	1.09e-07	0.0107	0.149	4.15e-07
	2	1.4e-07	1.02e-06	0.952	1.4e-07	1.02e-06	0.952	2.06e-08	1.09e-07	1	6.97e-08	1.24e-05	0.737

Kruskal-Wallis for Urea

	BIRCH			HC			SOM			k-Means			
	0	1	2	0	1	2	0	1	2	0	1	2	
BIRCH	0	1	0.079	6.75e-10	1	0.079	6.75e-10	0.136	0.593	5.01e-09	0.128	0.967	3.57e-09
	1	0.079	1	6.68e-09	0.079	1	6.68e-09	0.733	0.0225	4.16e-08	0.739	0.0971	3.46e-08
	2	6.75e-10	6.68e-09	1	6.75e-10	6.68e-09	1	2.02e-09	1.38e-10	0.449	1.21e-09	7e-09	0.614
HC	0	1	0.079	6.75e-10	1	0.079	6.75e-10	0.136	0.593	5.01e-09	0.128	0.967	3.57e-09
	1	0.079	1	6.68e-09	0.079	1	6.68e-09	0.733	0.0225	4.16e-08	0.739	0.0971	3.46e-08
	2	6.75e-10	6.68e-09	1	6.75e-10	6.68e-09	1	2.02e-09	1.38e-10	0.449	1.21e-09	7e-09	0.614
SOM	0	0.136	0.733	2.02e-09	0.136	0.733	2.02e-09	1	0.0418	1.19e-08	0.992	0.156	9.76e-09
	1	0.593	0.0225	1.38e-10	0.593	0.0225	1.38e-10	0.0418	1	1.1e-09	0.038	0.646	8.15e-10
	2	5.01e-09	4.16e-08	0.449	5.01e-09	4.16e-08	0.449	1.19e-08	1.1e-09	1	7.27e-09	5.08e-08	0.811

Kruskal-Wallis for AvgBovineLifetimeMilkYield

	BIRCH			HC			SOM			k-Means			
	0	1	2	0	1	2	0	1	2	0	1	2	
BIRCH	0	1	0.112	1.14e-07	1	0.112	1.14e-07	0.387	0.758	8.39e-09	0.204	0.796	4.8e-08
	1	0.112	1	9.92e-08	0.112	1	9.92e-08	0.335	0.0656	3.96e-09	0.662	0.0766	3.59e-08
	2	1.14e-07	9.92e-08	1	1.14e-07	9.92e-08	1	7.24e-09	8.36e-08	0.756	3.03e-08	1.21e-07	0.792
HC	0	1	0.112	1.14e-07	1	0.112	1.14e-07	0.387	0.758	8.39e-09	0.204	0.796	4.8e-08
	1	0.112	1	9.92e-08	0.112	1	9.92e-08	0.335	0.0656	3.96e-09	0.662	0.0766	3.59e-08
	2	1.14e-07	9.92e-08	1	1.14e-07	9.92e-08	1	7.24e-09	8.36e-08	0.756	3.03e-08	1.21e-07	0.792
SOM	0	0.387	0.335	7.24e-09	0.387	0.335	7.24e-09	1	0.248	1.27e-10	0.588	0.268	1.77e-09
	1	0.758	0.0656	8.36e-08	0.758	0.0656	8.36e-08	0.248	1	6.97e-09	0.125	0.956	3.56e-08
	2	8.39e-09	3.96e-09	0.756	8.39e-09	3.96e-09	0.756	1.27e-10	6.97e-09	1	7.52e-10	1.28e-08	0.974

Kruskal-Wallis for AvgBovineAge

	BIRCH			HC			SOM			k-Means			
	0	1	2	0	1	2	0	1	2	0	1	2	
BIRCH	0	1	0.184	3.33e-05	1	0.184	3.33e-05	0.636	0.704	4.38e-06	0.354	0.793	2.27e-05
	1	0.184	1	4.83e-05	0.184	1	4.83e-05	0.234	0.436	3.11e-06	0.57	0.365	2.92e-05
	2	3.33e-05	4.83e-05	1	3.33e-05	4.83e-05	1	2.05e-06	0.00025	0.971	1.32e-05	0.000176	0.812
HC	0	1	0.184	3.33e-05	1	0.184	3.33e-05	0.636	0.704	4.38e-06	0.354	0.793	2.27e-05
	1	0.184	1	4.83e-05	0.184	1	4.83e-05	0.234	0.436	3.11e-06	0.57	0.365	2.92e-05
	2	3.33e-05	4.83e-05	1	3.33e-05	4.83e-05	1	2.05e-06	0.00025	0.971	1.32e-05	0.000176	0.812
SOM	0	0.636	0.234	2.05e-06	0.636	0.234	2.05e-06	1	0.982	5.23e-08	0.534	0.918	8.2e-07
	1	0.704	0.436	0.00025	0.704	0.436	0.00025	0.982	1	5.68e-05	0.685	0.906	0.000208
	2	4.38e-06	3.11e-06	0.971	4.38e-06	3.11e-06	0.971	5.23e-08	5.68e-05	1	5.54e-07	3.77e-05	0.826

Kruskal-Wallis for InsemSuccessFactor

	BIRCH			HC			SOM			k-Means			
	0	1	2	0	1	2	0	1	2	0	1	2	
BIRCH	0	1	0.891	0.00693	1	0.891	0.00693	0.935	0.786	0.065	0.714	0.599	0.015
	1	0.891	1	0.000681	0.891	1	0.000681	0.936	0.647	0.0144	0.746	0.446	0.00163
	2	0.00693	0.000681	1	0.00693	0.000681	1	0.000807	0.0208	0.333	0.000308	0.039	0.694
HC	0	1	0.891	0.00693	1	0.891	0.00693	0.935	0.786	0.065	0.714	0.599	0.015
	1	0.891	1	0.000681	0.891	1	0.000681	0.936	0.647	0.0144	0.746	0.446	0.00163
	2	0.00693	0.000681	1	0.00693	0.000681	1	0.000807	0.0208	0.333	0.000308	0.039	0.694
SOM	0	0.935	0.936	0.000807	0.935	0.936	0.000807	1	0.68	0.0171	0.689	0.478	0.00199
	1	0.786	0.647	0.0208	0.786	0.647	0.0208	0.68	1	0.139	0.503	0.8	0.043
	2	0.065	0.0144	0.333	0.065	0.0144	0.333	0.0171	0.139	1	0.00715	0.225	0.535

Kruskal-Wallis point sums for 16 features:

For each cluster in both methods check if there is a single corresponding cluster in the other method.

If row/column has a $hvalue \geq 0.01$, add $1/12$ to the method-to-method sum; if a $hvalue \geq 0.01$ in every row/col, sum = 0.5 for a feature.

If row/column has a $hvalue \geq 0.05$, add another $1/12$ to the method-to-method sum; if a $hvalue \geq 0.05$ in every row/col, sum = 1 for a feature.

	BIRCH	HC	SOM	k-Means
BIRCH	16	16	10.3	11.3
HC	16	16	10.3	11.3
SOM	10.3	10.3	16	15
k-Means	11.3	11.3	15	16

APPENDIX D: KRUSKAL-WALLIS H-TESTS FOR HEALTH TREATMENT VARIABLE DATASET'S CLUSTERS

Kruskal-Wallis for TreatFertilityAvg

	BIRCH		HC		SOM		k-Means		
	0	1	0	1	0	1	0	1	
BIRCH	0	1	6.99e-12	1.99e-06	1.14e-57	7.54e-17	5.03e-51	4.55e-11	1.63e-65
	1	6.99e-12	1	7.56e-12	9.51e-09	8.61e-12	2.59e-10	7.86e-12	1.6e-09
HC	0	1.99e-06	7.56e-12	1	3.23e-73	1.4e-05	6.71e-80	0.0347	4.53e-90
	1	1.14e-57	9.51e-09	3.23e-73	1	4.12e-71	7.96e-16	1.39e-73	0.000553
SOM	0	7.54e-17	8.61e-12	1.4e-05	4.12e-71	1	2.06e-89	0.0135	7.61e-89
	1	5.03e-51	2.59e-10	6.71e-80	7.96e-16	2.06e-89	1	1.19e-88	5.53e-10

Kruskal-Wallis for TreatMetabolismAvg

		BIRCH		HC		SOM		k-Means	
		0	1	0	1	0	1	0	1
BIRCH	0	1	1.35e-05	0.0227	1.12e-15	2.02e-07	6.88e-21	0.00403	4.8e-15
	1	1.35e-05	1	1.9e-06	0.122	8.94e-08	0.0524	1.13e-06	0.0472
HC	0	0.0227	1.9e-06	1	5.88e-20	0.00302	2.49e-27	0.531	7.06e-20
	1	1.12e-15	0.122	5.88e-20	1	1.54e-26	0.344	4.69e-21	0.315
SOM	0	2.02e-07	8.94e-08	0.00302	1.54e-26	1	5.04e-37	0.0204	2.13e-27
	1	6.88e-21	0.0524	2.49e-27	0.344	5.04e-37	1	6.71e-29	0.878

Kruskal-Wallis for TreatDigestionAvg

		BIRCH		HC		SOM		k-Means	
		0	1	0	1	0	1	0	1
BIRCH	0	1	0.00203	0.119	1.16e-07	0.0101	4.3e-06	0.0778	2.42e-06
	1	0.00203	1	0.000457	0.363	0.000121	0.12	0.000352	0.194
HC	0	0.119	0.000457	1	6.26e-10	0.28	1.53e-08	0.817	1.33e-08
	1	1.16e-07	0.363	6.26e-10	1	1.25e-11	0.148	3.34e-10	0.388
SOM	0	0.0101	0.000121	0.28	1.25e-11	1	2.63e-10	0.398	2.76e-10
	1	4.3e-06	0.12	1.53e-08	0.148	2.63e-10	1	7.96e-09	0.565

Kruskal-Wallis for TreatUdderAvg

	BIRCH		HC		SOM		k-Means		
	0	1	0	1	0	1	0	1	
BIRCH	0	1	0.0227	0.0276	1.05e-12	1.68e-10	1.5e-27	0.00142	1.26e-15
	1	0.0227	1	0.00569	0.581	0.00011	0.238	0.00262	0.562
HC	0	0.0276	0.00569	1	1.53e-16	4.04e-05	7.31e-34	0.327	3.02e-20
	1	1.05e-12	0.581	1.53e-16	1	2.53e-27	0.15	1.1e-18	0.938
SOM	0	1.68e-10	0.00011	4.04e-05	2.53e-27	1	2.11e-51	0.00187	4.23e-33
	1	1.5e-27	0.238	7.31e-34	0.15	2.11e-51	1	2.19e-37	0.151

Kruskal-Wallis for TreatHoofAvg

	BIRCH		HC		SOM		k-Means		
	0	1	0	1	0	1	0	1	
BIRCH	0	1	0.0332	0.107	1.21e-07	6.25e-07	9.78e-17	0.0051	4.81e-12
	1	0.0332	1	0.0133	0.866	0.000709	0.728	0.00551	0.839
HC	0	0.107	0.0133	1	8.13e-10	0.000945	3.25e-20	0.249	7.64e-15
	1	1.21e-07	0.866	8.13e-10	1	8.91e-17	0.204	5.51e-12	0.372
SOM	0	6.25e-07	0.000709	0.000945	8.91e-17	1	2.88e-31	0.0295	1.04e-23
	1	9.78e-17	0.728	3.25e-20	0.204	2.88e-31	1	7.32e-24	0.737

Kruskal-Wallis for TreatRespiratoryAvg

		BIRCH		HC		SOM		k-Means	
		0	1	0	1	0	1	0	1
BIRCH	0	1	0.11	0.261	0.00038	0.00514	1.18e-05	0.0572	1.01e-05
	1	0.11	1	0.0587	0.786	0.0141	0.779	0.0339	0.878
HC	0	0.261	0.0587	1	2.43e-05	0.09	3.15e-07	0.44	3.23e-07
	1	0.00038	0.786	2.43e-05	1	7.2e-08	0.985	2.1e-06	0.748
SOM	0	0.00514	0.0141	0.09	7.2e-08	1	2.37e-10	0.351	2.65e-10
	1	1.18e-05	0.779	3.15e-07	0.985	2.37e-10	1	1.37e-08	0.713

Kruskal-Wallis for TreatCalfAvg

		BIRCH		HC		SOM		k-Means	
		0	1	0	1	0	1	0	1
BIRCH	0	1	0.144	0.223	0.000144	0.0066	1.51e-05	0.0705	1.97e-05
	1	0.144	1	0.0774	0.961	0.0252	0.858	0.053	0.962
HC	0	0.223	0.0774	1	6.42e-06	0.124	2.64e-07	0.55	4.87e-07
	1	0.000144	0.961	6.42e-06	1	4.06e-08	0.735	1.05e-06	0.996
SOM	0	0.0066	0.0252	0.124	4.06e-08	1	5.47e-10	0.343	1.45e-09
	1	1.51e-05	0.858	2.64e-07	0.735	5.47e-10	1	2.68e-08	0.715

Kruskal-Wallis point sums for 7 features:

For each cluster in both methods check if there is a single corresponding cluster in the other method.

If row/column has a $hvalue \geq 0.01$, add $1/12$ to the method-to-method sum; if a $hvalue \geq 0.01$ in every row/col, sum = 0.5 for a feature.

If row/column has a $hvalue \geq 0.05$, add another $1/12$ to the method-to-method sum; if a $hvalue \geq 0.05$ in every row/col, sum = 1 for a feature.

	BIRCH	HC	SOM	k-Means
BIRCH	7	5.5	3.5	4.25
HC	5.5	7	4.5	6.25
SOM	3.5	4.5	7	5.25
k-Means	4.25	6.25	5.25	7